The Thesis Committee for Sankar Bharathi Rengarajan
Certifies that this is the approved version of the following thesis:

A Method for Parameter Estimation and System Identification for Model Based Diagnostics

APPROVED BY
SUPERVISING COMMITTEE:

Supervisor:

Michael D. Bryant

Maruthi R. Akella

A Method for Parameter Estimation and System Identification for Model Based Diagnostics

by

Sankar Bharathi Rengarajan, B.E.

Thesis

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Master of Science in Engineering

The University of Texas at Austin

December 2010

Dedication

To my parents for their love, support and encouragement.

Acknowledgements

I would like to express my sincere gratitude to my advisor Dr. Michael Bryant, who introduced me to this wonderful field of Diagnostics. None of this work would have been possible without his technical expertise, vision and encouragement. I will always cherish the wonderful discussions we had on various issues, technical as well as personal. I would also like to thank my co-advisor, Dr. Akella Maruthi, who complemented this work with valuable inputs in his area of expertise and carefully reviewed this thesis.

I would also like to express my gratitude to Dr. Benito Fernandez for his significant inputs in conceptualizing this work and a person, whom I could always approach with a research problem or an idea. I would also like to thank Dr. Mitch Pryor and Dr. Paul Bommer for awarding me Teaching Assistantships, which were some of the best experiences in my journey at UT.

Special thanks to my lab colleagues, Jaewon Choi and Mohsen Nakhaeinejad for the wonderful discussions and friendly working atmosphere during my stay here at UT. Finally, I would like to thank my parents for their undying love and support. This work is dedicated to them.

December, 2010

Abstract

A Method for Parameter Estimation and System Identification for Model Based Diagnostics

Sankar Bharathi Rengarajan, M.S.E

The University of Texas at Austin, 2010

Supervisor: Michael D. Bryant

Model based fault detection techniques utilize functional redundancies in the static and dynamic relationships among system inputs and outputs for fault detection and isolation. Analytical models based on the underlying physics of the system can capture the dependencies between different measured signals in terms of system states and parameters. These physical models of the system can be used as a tool to detect and isolate system faults. As a machine degrades, system outputs deviate from desired outputs, generating residuals defined by the error between sensor measurements and corresponding model simulated signals. These error residuals contain valuable information to interpret system states and parameters. Setting up the measurements from a faulty system as baseline, the parameters of the idealistic model can be varied to minimize these residuals. This process is called "Parameter Tuning". A framework to

automate this “Parameter Tuning” process is presented with a focus on DC motors and 3-phase induction motors. The parameter tuning module presented is a multi-tier module which is designed to operate on real system models that are highly non-linear. The tuning module combines artificial intelligence techniques like Quasi-Monte Carlo (QMC) sampling (Hammersley sequencing) and Genetic Algorithm (Non Dominated Sorting Genetic Algorithm) with an Extended Kalman filter (EKF), which utilizes the system dynamics information available via the physical models of the system. A tentative Graphical User Interface (GUI) was developed to simplify the interaction between a machine operator and the module. The tuning module was tested with real measurements from a DC motor. A simulation study was performed on a 3-phase induction motor by suitably adjusting parameters in an analytical model. The QMC sampling and genetic algorithm stages worked well even on measurement data with the system operating in steady state condition. But the downside was computational expense and inability to estimate the parameters online – ‘batch estimator’. The EKF module enabled online estimation where update was made based on incoming measurements. But observability of the system based on incoming measurements posed a major challenge while dealing with state estimation filters. Implementation details and results are included with plots comparing real and faulty systems.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

1.1 MOTIVATION & OBJECTIVES

Reliability is the ability of a component, process or a system to perform a required task correctly under some stated condition, for a given period of time. Reliability depends on failures and faults [1]. Critical to industry, are effective techniques to increase reliability by diagnosing faults during inception, to allow for preventive measures. Early detection and precise isolation of faults reduce machine downtime, maintenance costs and increase safety. Maintenance costs can accrue to purchase prices of machines within a year of operation; downtime losses can far exceed this in minutes [2]. Techniques typically used by industry [3] to maintain operation and reduce damage of a machine include:

1. *Monitoring* - Measurable variables are compared to threshold values. Alarms are generated if variables exceed limits.
2. *Automatic Protection* – In case of a dangerous process state, corresponding to the measured state, an appropriate counteraction is taken.
3. *Supervision with fault diagnosis* – Based on measured variables, system features are calculated, and a fault diagnosis is performed. Suitable counteractions are made.

Though methods (1) and (2) are usually preferred owing to simplicity and reliability, an in-depth diagnosis of the system condition is not feasible with these techniques. Advanced methods (3) to perform supervision and fault diagnosis are needed to detect and isolate faults in terms of system states and parameters.

Most industrial diagnostic tools are signal-based and key on very specific features of waveforms and spectra. However, since physics, process dynamics and operating conditions vary with machines and change over time; these features are not reliable [3]. Classification techniques aim to learn the relationship between an input vector $'S'$ and output $'F'$ for a faulty system through extensive experimentation, thereby forming an explicit knowledge database of fault patterns. Needed are generic techniques, not system specific, but applicable on a wide range of systems.

Contrary to signal based methods, model based fault detection techniques allow a deep insight into the system behavior. The task consists of detection of faults using dependencies between different measured signals [4], predicted by mathematical models of the system or process. Well developed model based techniques provide advantages such as [3]:

- Early detection of small faults with abrupt change of incipient time behavior.
- Diagnosis of faults in actuator, process components or sensors, at the same time.
- Detection of faults in closed loops.
- Supervision of the process in transient states.

Lee & Bryant [5] introduced physics based models as a tool for fault diagnosis on induction motor and pumps, with information theory as a health metric. This technique is extended in this work to detect, isolate and assess machine faults, with application to diagnostics of 3-phase induction motors. As a machine degrades, system outputs deviate from ideal model outputs generating residuals defined by the error between sensor measurements and model simulated sensor signals. These error residuals contain information about current system states and parameters.

This work establishes a framework to utilize state space dynamic models for fault diagnostics, with emphasis on parameter estimation and system identification. The framework is suitable for application to any system, and is not limited to the 3-phase induction motor system described in this study.

1.2 METHODOLOGY

The first step to implement this model based fault diagnosis technique is to develop physical models that can replicate system response and accurately capture variations corresponding to individual system states and parameters. The induction motor model developed by Kim & Bryant [6] was used to stage faults in this study. The next step is to minimize the residuals generated from a faulty motor model compared to a healthy / nominal model. Setting up the measurements from the faulty system as base line, the parameters of the healthy model can be varied to minimize these residuals. This process is called "Parameter Tuning". The induction motor model is highly non-linear, similar to most real systems. The parameter tuning module in this study is designed to operate on highly non-linear systems.

The parameter tuning module is a multi-tier module that combines artificial intelligence techniques such as Quasi-Monte Carlo (QMC) sampling and Genetic Algorithm (Non-Dominated Sorting Genetic Algorithm – NSGA II) with an Extended Kalman filter (EKF), which utilizes the system dynamics information available via the physical models of the system. The initial stages (QMC & NSGA-II) operate on a batch of measurement data measured from the system. The EKF, which is a sequential estimator, then updates itself consistently based on incoming measurements. The initial stages help set up the Kalman filter, and can be used periodically to verify the filter

output and control the divergence of the extended Kalman filter. A tentative graphical user interface (GUI) was developed to permit a machine operator to select specific sensor outputs, desired parameters and specific faults, to tailor the diagnostic system to the user's needs.

Chapter 2 reviews existing literature on model based and signal based fault diagnostics techniques, and parameter estimation techniques.

Chapter 3 explains the induction motor model and state equations used to simulate the motor response. A one-to-one correspondence between the parameters of the model and real system parameters is established, to promote better understanding of a staged fault.

Chapter 4 lays theoretical and mathematical foundations for the parameter tuning module. The structure of the tuning module is explained, and reasons provided for each stage of the tuning module.

In Chapter 5, the proposed parameter tuning module is tested with real measurements from a DC motor. A simulation study was performed on the 3-phase induction motor model. Faults were induced by suitably adjusting model parameters. To detect the faults, parameters were estimated. Implementation details and results are included with plots comparing real and faulty systems.

Chapter 6 introduces the graphical interface built for the study and summarizes the work. Conclusions, limitations and scope for future research are also presented.

Chapter 2: Literature Review

The chapter summarizes existing literature in the field of model based diagnostics and parameter estimation. A number of publications have been reviewed and classified into five categories – (1) Model based & signal based fault diagnostics techniques (2) Artificial Intelligence (AI) in Model Based techniques (3) Parameter estimation using estimation filters & AI techniques (4) Quasi- Monte Carlo sampling techniques and (5) Approach taken in this work.

In Fault Detection & Isolation (FDI), models are based on the analytical (functional) redundancy. Inherent redundancy in the static and dynamic relationships among system inputs and outputs were exploited in FDI via mathematical models [7]. Various approaches to FDI using mathematical models include detection filter [8] [9]; innovation test using a single Kalman filter [4] or banks of Kalman filters or Luenberger observers [10], [4]; the parity space approach [11]; the parameter estimation technique [12]; and expert system applications [13]. A more detailed comprehensive analysis of each technique can be found in [14], and Basseville and Benveniste [15] and Patton et al. [13]. Knowledge- based models for fault diagnostics was introduced by Tzafestas in [16] using an on-line expert system which complements analytical model-based methods with a data-base of rules, current states, and an inference engine to make logical decisions.

Monitoring and fault detection in machines have moved in recent years from traditional techniques to Artificial Intelligence (AI) techniques [17]. The diagnostic procedure using such AI based models employ – (1) Fault signature extraction (2) Fault identification and (3) Fault severity evaluation. Some of the AI techniques typically include expert systems [17], artificial neural networks [18], fuzzy logic [19] and genetic algorithms [20] [17] etc.

A general approach to model based FDI involves residual generation and assessment based on these residuals. Early contributions to the parity space approach were made by [21], [22], [23] and [24]. The key idea is to check consistency of mathematical equations using actual measurements. Once pre-assigned error bounds are exceeded, a fault condition is defined [10]. Mehra et al [25] and Frank et al [26] approached the FDI problem based on single or banks of Luenberger observers or Kalman filters. The fault detection filter proposed by Beard [8] and Jones [9] required very precise modeling and does not account for parameter variations, measurement noise and disturbances [27]. The parameter identification approach is to detect faults via estimation of parameters of the mathematical model along with states [3]. This approach helps to detect as well as isolate faults.

Extensive literature employs parameter estimation techniques used in fault diagnosis. Young [28] reviewed parameter estimation techniques developed for continuous time models over the period 1958-1980. Cardozo et al [29] described a distributed expert system written in a rule based language (OPS5) for fault diagnosis of a power system network. Bishop [20] applied genetic algorithms to identify parameters of an induction machine. Comly et al [30] developed a fuzzy inferencing test bed to investigate faults in diverse engineering products. Julian Luis et al [31] proposed a set of on-line estimation algorithms based on linear regression models for motor faults. Said et al [32] dealt with broken rotor bar detection in induction motors using an extended Kalman filter. Jack and Nandi [33] examined performance of machine learning algorithms such as Artificial Neural Networks and Support Vector Machines in combination with a genetic algorithm to classify the parameter space into faulty/ no-faulty regions. Samanta et al [34] presented an ANN technique for fault diagnosis on rolling-element bearings using time-domain vibration signals of rotating machinery with

healthy and faulty bearings as input features. Kalyanmoy Deb et al [35] presented a fast and elitist multi-objective genetic algorithm – Non Dominating Sorting Genetic Algorithm (NSGA-II) that is efficient solving highly non-linear problems with a multi-dimensional parameter space. Rasmus and Vadstrup [36] used differential evolution technique for parameter identification on an induction motor. Smail Bachir et al [37] presented a new model to estimate squirrel-cage induction motors under stator and rotor faults using extended Kalman filters. Messaoudi et al [38] built a robust non-linear observer for states and parameter estimation in sensorless induction motor drives.

"Design of Experiments" (DOE) chooses a set of samples in the design space to gain maximum information using a limited number of samples. Sampling techniques developed exclusively for multi-dimensional integration include the Quasi-Monte Carlo based Hammersley sampling and Halton sequencing [39]. Woo et al [40] used Hammersley sampling to capture surface roughness variations on a machined surface, and showed that random sampling does not capture the variations sufficiently. An application to offline quality control of a continuous-stirred tank reactor by Kalagnanam et al [41] showed that variations of the parameter distributions could be captured by Hammersley sampling uses 40 times fewer points than a pure random sampling.

Here, a model-based diagnostic module will be presented, that uses features of the above mentioned techniques and mathematical models coupled with parameter estimation algorithms to detect and isolate faults. The parameter values will be bounded using designer specifications of the system. Hammersley sampling technique will first be employed to capture the variations of a cost function in the parameter design space. A genetic algorithm (NSGA-II) will then find the region that most likely contains the global optimum in the multi-dimensional space. Finally, an extended Kalman filter (EKF) that accounts for measurement and process noise estimates system states and parameters

using dynamic mathematical models and new incoming measurements. The EKF permits real time diagnostic monitoring.

Chapter 3: Modeling of Physical System

The proposed architecture for fault diagnostics relies on physics based models that replicate ideal (healthy) system behavior. In this study, model based diagnostics was used to detect and isolate faults in a squirrel cage induction motor. A simulation based study employed the induction motor model by Kim & Bryant [6].

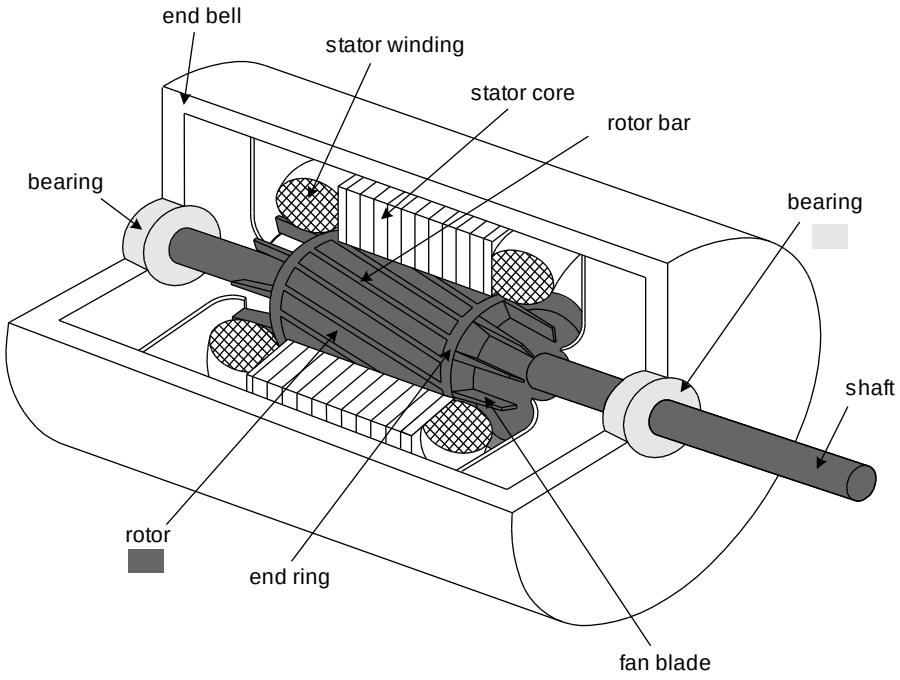

Figure 2.1: Exploded view of a squirrel cage induction motor

2.1 Squirrel Cage Induction Motor Model

A motor has two major sub-systems: a rotating rotor and a non-rotating stator. Squirrel cage induction motors, widely used for variable speed systems, are simple, rugged and inexpensive. Figure 2.1 depicts a squirrel cage induction motor. Induction

motor models available in literature typically use state-space two reaction theory by Park [42] including Ghosh and Bhadra's bond graph model [43]. Here an induction machine, represented by the mutually perpendicular $\alpha - \beta$ model in a stationary reference frame is linked to three phase current (voltage) source by a power conserving transformation. Kim & Bryant [6] modified the existing bond graph model to form a one-to-one correspondence between the motor components and bond graph elements. This facilitates simulation of faults by suitably adjusting parameter values inside the model and observing system response. The final bond graph model, shown in Figure 2.2, classified the motor system into various energy domains – electrical, magnetic and mechanical.

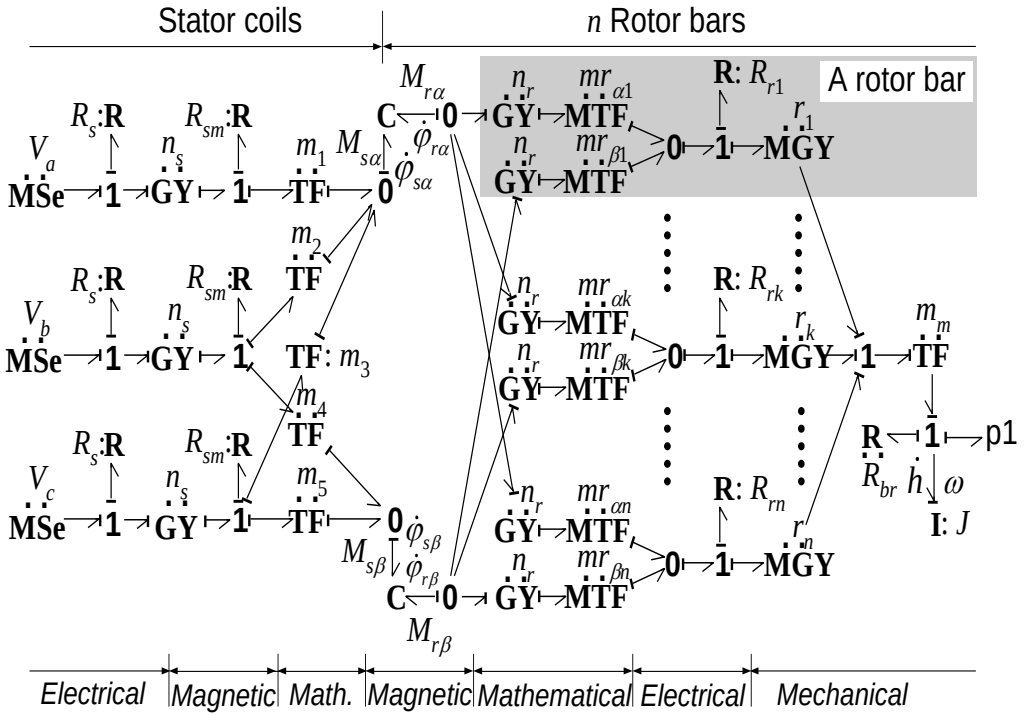

Figure 2.2: Bond graph model of a 3-phase squirrel cage induction motor [6]

When energized by an AC voltage supply, the stator coils form a magnetic field vector rotating about the central axis of the stator. This time varying field passes through the individual rotor bars and induces a voltage over the rotor bars. Resulting bar currents follow the path: bar → end ring → opposite side bar → opposite side end ring → original bar. This time varying current loop produces a magnetic field that tries to align itself with the stator field. But the stator field rotates and the rotor follows the stator field. This is motor action. The speed of the motor depends on the speed of the rotating magnetic field.

$Mse: V_a$, $Mse: V_b$ and $Mse: V_c$ represent the three-phase supply voltage, and stator coil resistances across the three phases are represented as R_{sa}, R_{sb} and R_{sc}. The magnetic losses in the stator windings are modeled as R_{sm} and the gyrator modulus $GY: n_s$ represents the number of turns in the stator coil. A transformation from a three phase a, b, c frame to two phase α, β frame was represented in matrix form in Hancock [44]. The battery of transformers $m_1 = \sqrt{3/2}$, $m_2 = m_3 = -\sqrt{6}$, $m_4 = \sqrt{2}$, and $m_5 = -\sqrt{2}$ represent elements of the transformation matrix.

$$\begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} = \begin{bmatrix} 1/m_1 & 1/m_2 & 1/m_3 \\ 0 & 1/m_4 & 1/m_5 \end{bmatrix} \begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix} = \begin{bmatrix} \sqrt{\frac{2}{3}} & -\frac{1}{\sqrt{6}} & -\frac{1}{\sqrt{6}} \\ 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix} \qquad (3.1)$$

The constitutive law for the two port $'C'$ element

$$\begin{bmatrix} M_{si} \\ M_{ri} \end{bmatrix} = \begin{bmatrix} a_i & b_i \\ b_i & c_i \end{bmatrix} \begin{bmatrix} \emptyset_{si} \\ \emptyset_{ri} \end{bmatrix} \qquad (3.2)$$

captures the interaction between the stator and rotor fields. The $'C'$ element relates the magneto-motive force M to flux $\emptyset$ via the reluctance matrix with elements,

$$a_i = \frac{{n_s}^2 L_{ri}}{L_{si}L_{ri} - {L_{mi}}^2}, b_i = \frac{-n_{si}n_r L_{mi}}{L_{si}L_{ri} - {L_{mi}}^2}, c_i = \frac{{n_r}^2 L_{si}}{L_{si}L_{ri} - {L_{mi}}^2} \quad (3.3)$$

where L_{si} and L_{ri} represent the self inductances of rotor and stator and L_{mi} is the mutual inductance between the stator and the rotor. The subscript $i = \alpha$ or β indicates the motor phase. To incorporate the effect of individual rotor bars into the bond graph, the rotor's α or β phase currents and voltages should be split into separate bar currents and voltages. The frames a, b, c and α, β are stationary with respect to the stator, but to the rotating rotor, individual bar currents depend on the angular displacement $'\theta'$ of each bar. Using results in Hancock

$$i_{rk} = i_{\alpha k}(mr_k) + i_{\beta k}(mr_{(k+n)}) \quad (3.4)$$

where $mr_{\alpha k}$ and $mr_{\beta k}$ relate the angular position of the rotor bar relative to the flux field, given by

$$mr_k = \sqrt{\frac{2}{n}}\cos\left(\theta + \frac{2(k-1)\pi}{n}\right), \; mr_{(k+n)} = \sqrt{\frac{2}{n}}\sin\left(\theta + \frac{2(k-1)\pi}{n}\right) \quad (3.5)$$

Here $k = 1,2 \ldots n$ represents the k^{th} rotor bar in a total of $'n'$ bars. This sum of currents is modeled by the $'0'$ junction with corresponding transformers with respect to each rotor bar. The resistive elements R_{rk} represent the losses in each rotor bar.

The modulated gyrators $MGY: r_k$ transform the rotor currents in each bar to the electro-magnetic torque contributed by each bar and governed by the relation,

$$r_k = n_r \emptyset_{r\beta}\, mr_k - n_r \emptyset_{r\alpha}\, mr_{(k+n)} \tag{3.6}$$

The final electro-magnetic torque, T_e for a P_p - pole machine is,

$$T_e = \frac{P_p}{2} \sum_{k=1}^{n} r_k i_{rk} \tag{3.7}$$

The transformer modulus $m_m = \frac{P_p}{2}$. Bearing friction effect is captured by the resistance $R: R_{br}$. The state equations extracted from the bond graph are

$$\dot{\emptyset}_{\alpha s} = \frac{1}{m_1 . n_s}\{V_a - \frac{R_{sa}}{{n_s}^2 + R_{sa}R_{sm}}(R_{sa}V_a + \frac{n_s}{m_1}M_{s\alpha})\}$$

$$+ \frac{1}{m_2 . n_s}\{V_b - \frac{R_{sb}}{{n_s}^2 + R_{sb}R_{sm}}(R_{sb}V_b + \frac{n_s}{m_2}M_{s\alpha} + \frac{n_s}{m_4}M_{s\beta})\}$$

$$+ \frac{1}{m_3 . n_s}\{V_c - \frac{R_{sc}}{{n_s}^2 + R_{sc}R_{sm}}(R_{sc}V_c + \frac{n_s}{m_3}M_{s\alpha} + \frac{n_s}{m_5}M_{s\beta})\}$$

$$\dot{\emptyset}_{\beta s} = \frac{1}{m_4 . n_s}\{V_b - \frac{R_{sb}}{{n_s}^2 + R_{sb}R_{sm}}(R_{sb}V_b + \frac{n_s}{m_2}M_{s\alpha} + \frac{n_s}{m_4}M_{s\beta})\}$$

$$+ \frac{1}{m_5 . n_s}\{V_c - \frac{R_{sc}}{{n_s}^2 + R_{sc}R_{sm}}(R_{sc}V_c + \frac{n_s}{m_3}M_{s\alpha} + \frac{n_s}{m_5}M_{s\beta})\}$$

$$\dot{\emptyset}_{\alpha r} = -\frac{M_{r\alpha}}{{n_r}^2}\sum_{k=1}^{n} m\,{r_k}^2.R_{rk} - \frac{M_{r\beta}}{{n_r}^2}\sum_{k=1}^{n} m\,r_k.mr_{(k+n)}R_{rk} - \frac{h}{m_m J}\sum_{k=1}^{n} m\,r_k.r_k$$

$$\dot{\emptyset}_{\beta r} = -\frac{M_{r\beta}}{{n_r}^2}\sum_{k=n+1}^{2n} m\,{r_k}^2.R_{r(k-n)} - \frac{M_{r\alpha}}{{n_r}^2}\sum_{k=n+1}^{2n} m\,r_k.mr_{(k-n)}R_{r(k-n)} - \frac{h}{m_m J}\sum_{k=n+1}^{2n} m\,r_k.r_{(k-n)}$$

$$\dot{h} = -\frac{M_{r\alpha}}{m_m n_r}\sum_{k=1}^{n} m\,r_k.r_k + \frac{M_{r\beta}}{m_m n_r}\sum_{k=1}^{n} mr_{(k+n)}.r_k - \frac{h}{J}R_{br}$$

$$\dot{\theta} = \frac{1}{m_m}.\frac{h}{J} \qquad (3.8)$$

where,

$\emptyset_{\alpha s}, \emptyset_{\alpha r}, \emptyset_{\beta s}$ and $\emptyset_{\beta r}$ - stator and rotor phase magnetic fluxes

h - rotor angular momentum

θ - shaft angular displacement.

Chapter 4: Parameter Tuning & System Identification

4.1 INTRODUCTION

The "Parameter Tuning" module is critical to the proposed Model Based Diagnostics & Prognostics architecture. Complex models developed previously perform "Hardware-in-the-Loop" simulations. As a machine degrades, system outputs deviate from desired outputs, generating residuals defined by the error between sensor measurements and corresponding signals simulated by the model. These error residuals contain valuable information to interpret system states and parameters. The tuning module simultaneously estimates system states and parameters to minimize error residuals. Once the error residuals are minimal, a direct one-to-one physical correspondence between the model's parameters and the real system elements simplifies fault classification.

The first step determines which faults to diagnose, and the corresponding parameters to track. The number of parameters defines the dimensionality of the parameter space. The next step is to set extremes for each parameter, using the designer's specifications for the system, or operator experience. These bounds define the admissible operating region for each parameter. After the admissible operating region of the parameter space has been defined, a deterministic sampling technique scans the entire operating region without bias towards any particular sub-region, which could possibly occur using a random sampling. At each sampling point, error residuals are calculated to identify regions where residuals are minimal. These 'hot spots' in the parameter space are likely operating points of the system. With the search region reduced, randomness is incorporated in the algorithm to maximize the likelihood of attaining a global minimum

for the error residuals. Then a "Non-Dominating Sorting Genetic Algorithm (NSGA)" [35] is run in small regions about the identified hot spots to locate the global minimum. These computations are done in parallel over the hot spots. The resulting values of parameters become initial values to an "Extended Kalman Filter" which by accounting for uncertainty due to sensor and process noise further reduces the error residuals and maximizes accuracy of identifying the parameters and operating condition of the system.

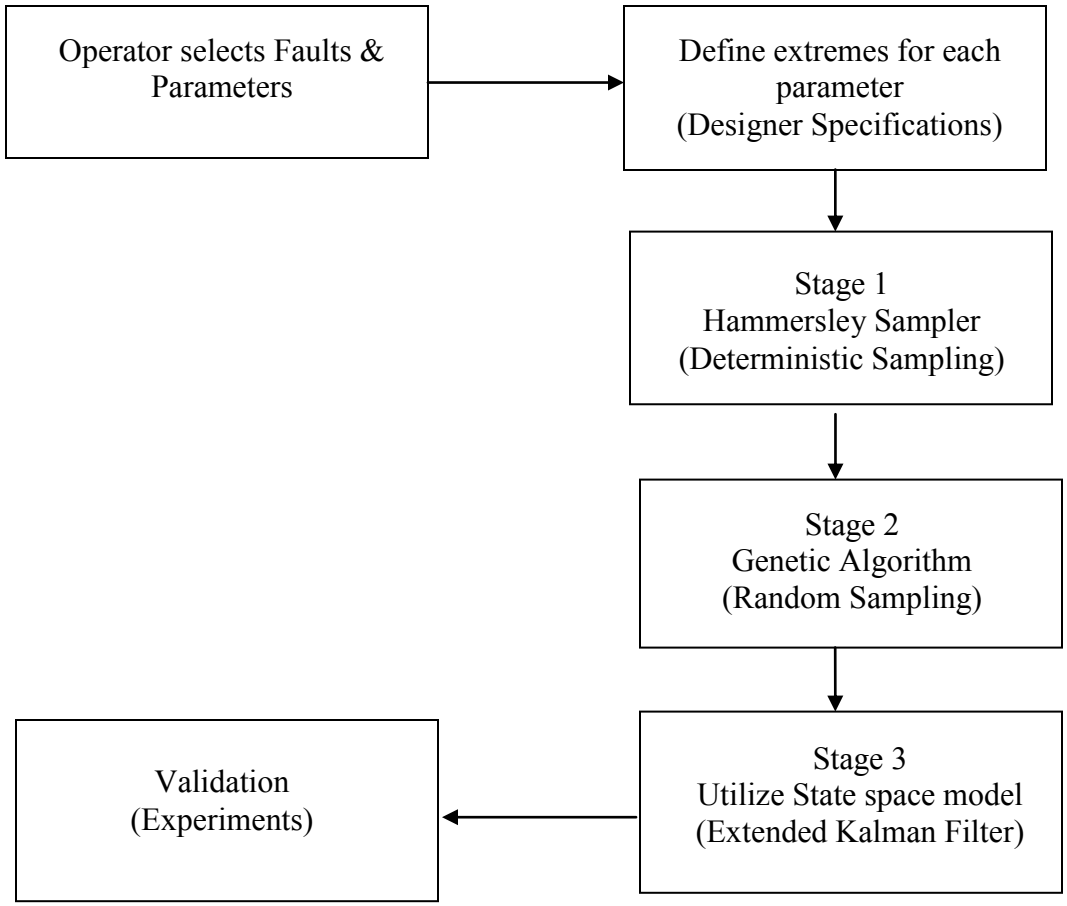

Figure 4.1 Schematic Sketch of the Parameter Tuning Module

Figure 4.1 shows the parameter tuning module structure. Since the models are highly non-linear, multiple regions can exist in the parameter space where error residuals have similar values i.e. local minima. The challenge is to identify correct values of the parameters, without getting trapped in local minima.

4.2 FRAMING THE COST FUNCTION – TOTAL ERROR RESIDUAL FUNCTION

Error residuals between model simulations and real system measurements will gauge the performance of the system. Since the error cost function depends upon the system outputs tracked, a 'Sensitivity' analysis and an 'Observability' analysis between measured output states and system parameters to be tuned is needed. Observability analysis indicates what combination of states and "when" (transient or steady-state), the measurements contain relevant information about parameter values. A sensitivity analysis indicates "what" system outputs can estimate a parameter accurately. With multiple outputs, the final "error residual function" is

$$Total\ error\ residual = (w_1 * r_1 + w_2 * r_2 + \cdots + w_n * r_n) \tag{4.1}$$

where $r_1, r_2, \ldots, r_n$ are the error residuals computed for the $'n'$ outputs tracked. Weights $w_1, w_2, \ldots., w_n$ are proportional to the sensitivity of a parameter to each output tracked. "Total Error Residual" is the objective function that must be minimized by varying the parameters $p_1, p_2, p_3 \ldots, p_n$ of the system i.e.

$$Minimize_{p_1, p_2, \ldots p_n}\ [Total\ Error\ Residual] \tag{4.2}$$

subject to constraints or bounds that have been set previously.

4.3 DETERMINISTIC SAMPLING – STAGE 1

From a statistical viewpoint, sampling techniques permit valid inferences about a system. The information that can be inferred about the system depends on:

1. Number of Sampling Points
2. Quantum of variations of the system output captured

Though the former maximizes the number of sample points, the latter can be controlled via an effective sampling strategy. As the number of parameters increases, dimensionality increases, making critical an effective sampling strategy to capture variations with fewer sample points and limited computational expense.

In a random sampling, sample points could lie very close to each other. Under these circumstances, variations of the cost function cannot be captured effectively. The deterministic sampler uniformly samples the parameter space, ensuring that the variations across all regions of the parameter space are captured, without bias to any region. The deterministic upper and lower bounds for any sequence of integration are expressed in terms of discrepancy - the deviation of the sequence from a uniform distribution [39]. Sampling techniques of low discrepancy measure are preferred. 'Quasi-Monte Carlo (QMC)' sampling techniques produce points 'almost' random, but better distributed than random samples. 'Halton Sequence Sampling' and 'Hammersley Sequence Sampling' are popular QMC sampling techniques used extensively in modern Design of Experiments (DOE) and Computational Image Acquisition [45]. Rafajlowicz [46], Simpson et al [47] and Lee et al [48] suggest that Hammersley Sequence technique has low discrepancy measure and ensures a uniform distribution of sample points, even in higher dimensions.

The algorithm is scalable. To add more sample points across a particular region in the parameter space suspected to be highly non-linear, the Hammersley technique can generate new sample points in that region without having to resample the entire

parameter space. Since model evaluation at each step is computationally expensive, minimizing the number of model evaluations is mandatory. Taking into account, the uniform spread and scalability, Hammersley sequence technique outperforms the Halton sequence for the deterministic sampling stage.

4.3.1 Hammersley's algorithm

Any integer $'n'$ can be expressed in radix $'R'$ notation (R is an integer that indicates the base of number $'n'$) as follows:

$$n = n_m n_{m-1} n_{m-2} n_{m-3} \dots\ n_2 n_1 n_0 \qquad (4.3)$$
$$n = n_0 + n_1 R + n_2 R^2 + \cdots + n_m R^m$$

where $m = \log_R n = [\ln n / \ln R]$. A unique fraction between 0 and 1, called the $inverse\ radix\ number$, can be constructed by reversing the order of the digits of $'n'$around the decimal point as follows:

$$\emptyset_R(\mathrm{n}) =. n_0 n_1 n_2 n_3 \dots\dots n_{m-1} n_m \qquad (4.4)$$
$$\emptyset_R(\mathrm{n}) \ = n_0 R^{-1} + n_1 R^{-2} + \cdots + n_m R^{-m-1}$$

Let $'k'$ be the dimension of the space to be sampled and $'N'$ be the desired number of sample points. Hammersley sample points for the $'k'$ dimensional space have co-ordinates in the space according to the sequence:

$$\vec{Z}_k\ (n) = \left(\frac{n}{N}, \emptyset_{R_1}(n)\ ,\ \emptyset_{R_2}(n)\ \ \dots,, \emptyset_{R_{k-1}}(n)\right) \qquad (4.5)$$

where $n = 1,2,3 \ldots . N$ and $R_1, R_2, R_3 \ldots . R_{k-1}$ are the first $'k-1'$ prime numbers such that $R_1 < R_2 < R_3 \ldots < R_{k-1}$. The final Hammersley sequence of points are given by the relation

$$\vec{X}_k\,(n) = I - \vec{Z}_k\,(n) \tag{4.6}$$

where $'I'$ is a vector of 1's with length equal to that of the dimensionality of the space $'k'$.

4.3.2 Structure of Hammersley Sampler

Extremes for parameters define the region over which error residuals are to be calculated. The amount of variation that can be captured in the cost function is directly proportional to the number of sample points used. If more computational resources are available, then more sample points in the parameter space are recommended, to gain maximum information about the variation of the Total Residual Error function. The Hammersley Sampler module, not a one-step process, gradually evolves over a fixed number of generations to narrow down the search region. The module is structured to maximize the likelihood of finding the global minima. Figure 4.2 depicts hotspots, defined as points with very low cost function values in a 3-dimensional parameter space. Since the algorithm runs these evaluations in parallel, the error residuals are calculated across various hotspots simultaneously, reducing computation time.

The deterministic sampler module follows the algorithm:

1. Generate Hammersley points based on $'k'$dimensions and $'N'$ sample points.
2. Simulate the system output at those Hammersley points and calculate error residuals.
3. Rank the top five points based on lowest values of error residuals. This constitutes a "progeny".
4. Define new regions around those points, and resample (repeat steps 1 to 3) within those regions.
5. If number of generations > 5, break the loop and pass the best 5 points across all progenies to the random sampling stage (stage 2).

Restricting the number of generations to 5 provides a region, neither too small nor too large for the Genetic Algorithm (stage 2) to operate. Figure 4.2 depicts the sampling algorithm for a case of 3 parameters which results in a parameter space of 3 dimensions. The algorithm can be extended to $'n'$dimensions.

4.4 RANDOM SAMPLING USING GENETIC ALGORITHM – STAGE 2

The five points from deterministic sampling with lowest "Total Error Residual" function become input parameters to the random sampling stage. New regions are defined about these sample points, where a Non-Dominating Sorting Genetic Algorithm, NSGA-II runs in parallel.

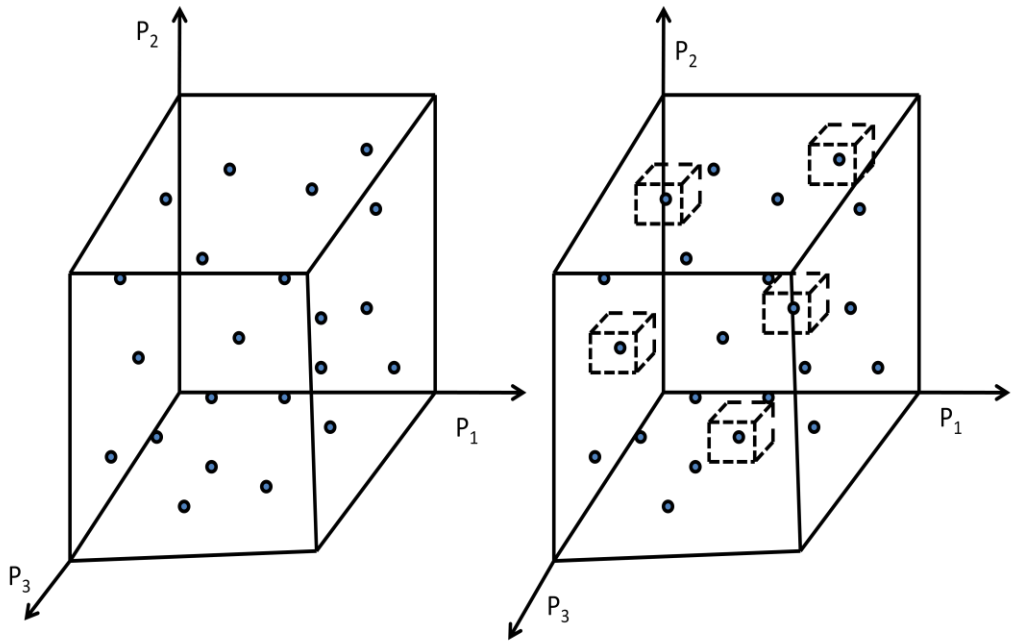

Figure 4.2 Hot spots identification in a 3-Dimensional Parameter Space

4.4.1 Random Sampling Module Description

The structure of the random sampler module is illustrated in Figure 4.3. The NSGA-II algorithm uses an exploratory and intelligent search algorithm because there is no gradient based information available for individual parameter degradation. The parameter variation is a slow process unlike system states that uniquely represent the system at any instant of time. The algorithm is effective solving multi-objective optimization problems [49]. Though multiple outputs are tracked and corresponding error residuals calculated, the Total Error Residual function is a scalar form of all the individual error residuals.

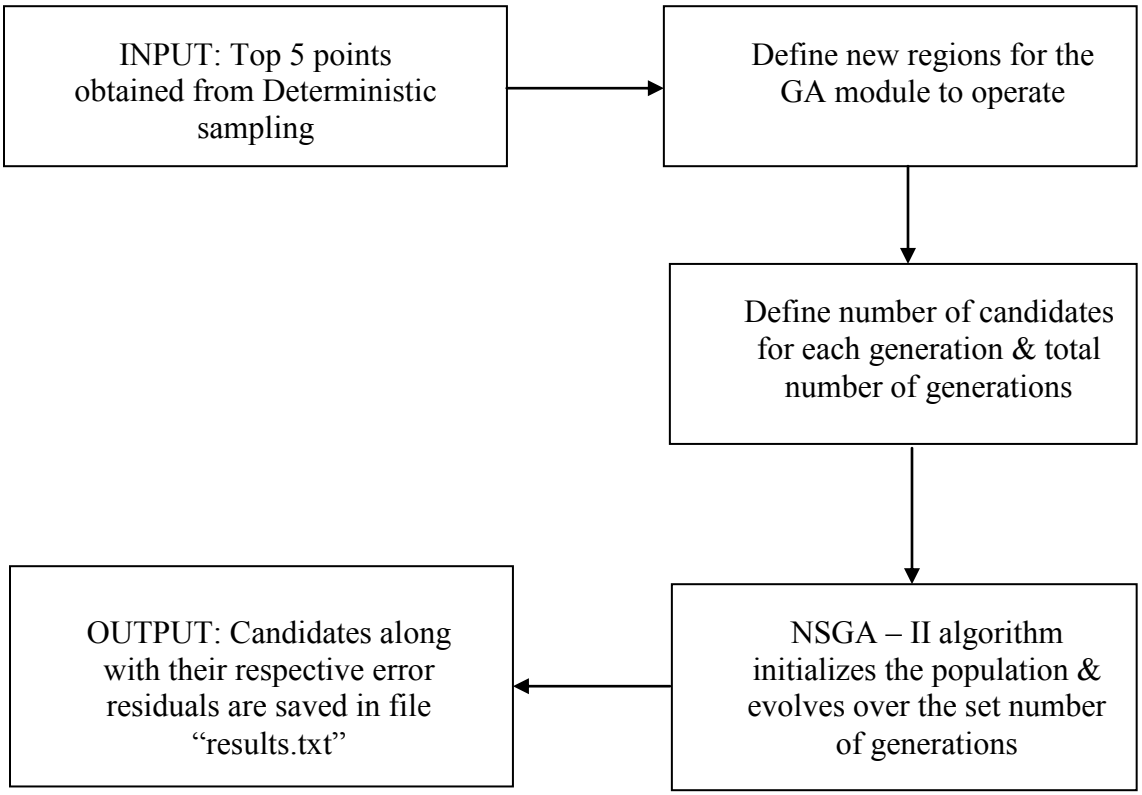

Figure 4.3 Schematic sketch of the GA module

The weights in Total Error Residual function (eq. 4.1) lend flexibility to the entire module. For example, if the operator suspects malfunction of a particular sensor, but prefers not to completely discard information from that sensor, the weights in the Total Error Residual function can be suitably adjusted. Weights need not be assigned purely based on the sensitivity of a parameter to the tracked output, but can also be used to assign a belief to the value of a particular sensor input.

4.4.2 Non-Dominating Sorting Genetic Algorithm (NSGA –II)

The NSGA –II, a non-domination based genetic algorithm for multi-objective optimization problems, has a good sorting algorithm, incorporates elitism, and needs no sharing parameter be chosen a priori.

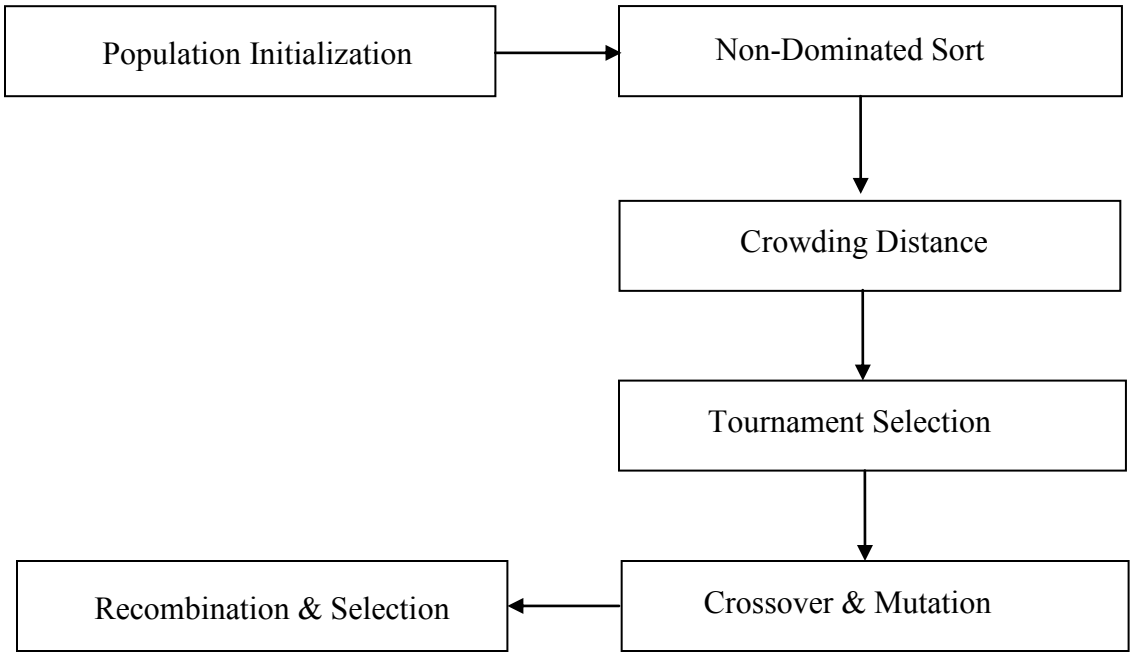

Figure 4.4 Schematic sketch of control flow of NSGA-II algorithm

4.4.2.1 Population Initialization

Error residuals for each tracked output become individual objective functions for the algorithm. System parameters chosen previously correspond to the total number of decision variables that the NSGA-II algorithm can vary to obtain a global minimum. Each chromosome for the algorithm is initialized based on minimum and maximum local

bounds for the parameters. These bounds arise from new regions defined around the best five points obtained from the deterministic sampling algorithm. A random number generator picks a number between the minimum and maximum possible values for each parameter. This serves as an initial population for the NSGA-II. Once the initial candidates have been generated, the algorithm evaluates the fitness function, which is the Total Error Residual function for these candidates.

4.4.2.2 Non-dominated sort

The initial population is sorted based on non-domination. An individual dominates another if its objective functions are no worse than the other, and at least one of its objective functions it is better than the other. The fast sort algorithm in Figure 4.4 utilizes the following information for each individual candidate:

(1) Number of individuals $'n_p'$ that dominate a particular candidate and

(2) The set $'S_p'$ that the individual candidate dominates

- For each individual $'p'$ in the main population $'P'$ do the following:
  - Initialize $S_p = \emptyset$. Set S_p would contain all individuals dominated by $'p'$
  - Initialize $n_p = 0$. $'n_p'$ is the number of individuals that dominate $'p'$
  - For each individual $'q'$ in $'P'$
    - If $'p'$ dominates $'q'$ then
      - Add $'q'$ to the set S_p, i.e., $S_p = S_p \cup \{q\}$
    - Else if $'q'$ dominates $'p'$ then
      - Increment the domination counter i.e. $n_p = n_p + 1$
  - If $n_p = 0$, it no individuals dominate $'p'$, and $'p'$ belongs to the first front. The rank of $'p'$ is set to 1
- Repeat for all individuals in main population $'P'$
- While $F_i \neq \emptyset$, i.e., a particular i^{th} front is not empty
  - Initialize a new set $'Q' = \emptyset$, to store candidates in $(i + 1)$ front.
  - For each individual $'p'$ in the front F_i
    - For each individual $'q'$ in S_p - individuals dominated by $'p'$
      - $n_q = n_q - 1$, decrement domination count for $'q'$
      - If $n_q = 0$, then none of individuals in the front would dominate $'q'$. Hence set rank of $'q'$ as $(i + 1)$
  - Increment the front counter by 1. Hence $'Q' = F_i$

Figure 4.5 Non-Dominated Sorting algorithm as described in [50]

4.4.2.3 Crowding Distance Calculation

Once the non-dominated sorting has been completed, crowding distance information for every individual candidate is assigned. Crowding distance is defined as the Euclidean distance between each individual in a front, based on a $'m'$ dimensional hyperspace. Crowding distance is assigned front-wise. Comparing crowding distance of two candidates in different fronts has no meaning. The crowding distance calculation is shown in Figure 4.5. Crowding distance of boundaries is set to infinity. Hence, boundaries are always included in the population set.

- For each front F_i, with $'n'$ individuals
  - Initialize distance to zero for all individuals i.e.$F_i(d_j) = 0$, where $'j'$ corresponds to the j^{th} individual in front F_i.
  - For each objective function $'m'$
    - Sort the individuals in front F_i , based on objective $'m'$
    - Assign infinite distance to boundary values for each individual in F_i i.e. $I(d_1) = \infty$ and $I(d_n) = \infty$
    - For $k = 2$ to $(n - 1)$
      - $I(d_1) = I(d_k) + \frac{I(k+1).m - I(k-1).m}{f_{max} - f_{min}}$

Figure 4.6 Calculating the crowding distance for individuals

4.4.2.4 Tournament Selection

The selection of individuals to be processed for crossover & mutation is based on the crowded – comparison operator $'\varepsilon'$ and a "Binary Tournament Selection". The basic mechanism for tournament selection consists of [51]:

- Randomly Choosing – with or without replacement, a predetermined number of individuals, and picking the best from these individuals – either probabilistically or deterministically.
- Repeat the above procedure 'N' times (N=population size) to fill the next generation of candidates.

In the case of the NSGA-II algorithm, the comparison between individuals is deterministic and done using the crowded comparison operator based on the following lemma:

1. Individuals in a particular front F_i will have rank $p_{rank} = i$.
2. Crowding distance of the candidate is $F_i(d_j)$
    - $p < q$ if
        - $p_{rank} < q_{rank}$ (or)
        - If both the candidates are from the same front, $F_i(d_p) > F_i(d_q)$. The candidate with higher crowding distance is preferred to maintain diversity in the population, and to avoid too many points that lie very close to one another.

4.4.2.5 Genetic Operators – Crossover & Mutation

Genetic Algorithms are either binary–coded or real–coded to represent individuals in a population. The NSGA-II algorithm is a real – coded GA, uses the "Simulated Binary Crossover" [52] and "Polynomial Mutation" for generating new offsprings and

retaining candidates from the current generation. The Simulated Binary Crossover algorithm defines children:

$$c_{1,k} = 0.5\left[(1-\beta_k)p_{1,k} + (1+\beta_k)p_{2,k}\right] \tag{4.7}$$
$$c_{2,k} = 0.5\left[(1+\beta_k)p_{1,k} + (1-\beta_k)p_{2,k}\right]$$

Here $c_{i,k}$ is the i^{th} child with k^{th} component, $p_{i,k}$ is the selected parent, and $\beta_k \geq 0$ is a sample from a random number generator having density

$$p(\beta) = 0.5[(\eta_c + 1)\beta^{\eta_c}, \qquad if\ 0 \leq \beta \leq 1 \tag{4.8}$$

$$p(\beta) = 0.5[(\eta_c + 1)\frac{1}{\beta^{\eta_c+2}}, \qquad if \quad \beta \geq 1$$

This distribution can arise from a uniformly sampled random number $'u'$ between (0, 1) Here η_c is the distribution index for cross-over. The cross-over index determines how well spread (or) diverse the children will be from their parents. Therefore,

$$\beta(u) = (2u)^{\frac{1}{\eta+1}} \tag{4.9}$$

$$\beta(u) = \frac{1}{2(1-u)^{\frac{1}{\eta+1}}}$$

Mutation is intended to capture regions in the parameter space that could be missed in the initial generations, but might encompass the global minima.

For generating candidates via mutation, the NSGA-II algorithm uses Polynomial mutation described by:

$$c_k = p_k + (p_k{}^u - p_k{}^l)\delta_k \tag{4.10}$$

Here c_k is the child, p_k is the parent with $p_k{}^u$ being the upper bound on the parent component, $p_k{}^l$ the lower bound, and δ_k is a small variation calculated via a polynomial distribution

$$\delta_k = (2r_k)^{\frac{1}{\eta_m+1}} - 1, \qquad if\ r_k < 0.5 \tag{4.11}$$

$$\delta_k = 1 - [2(1 - r_k)]^{\frac{1}{\eta_m+1}}, \qquad if\ r_k \geq 0.5$$

In eq. 4.11, r_k is a uniformly sampled random number between (0, 1), and η_m is the mutation distribution index. Mutation rate is kept low compared to crossover rate, to reduce the risk of losing good candidates from the present generation.

4.4.2.6 Recombination & Selection

The offsprings generated by the mutation and crossover operators are combined with the current generation. Selection is taken over the entire set. Since offsprings are added to the best individuals in the current generation, elitism in the population is ensured. The process repeats for subsequent generations.

4.4.2.7 Why Genetic Algorithm?

For non-linear functions, there exists a possibility that various combinations of system parameters could render identical error residual values. To compensate, the Genetic Algorithm module can incorporate human intelligence into the parameter tuning

module. The module presents the operator with various possible operating conditions of the system. An experienced or intelligent operator of the machine can eliminate certain combinations of parameters based on other external criteria. The points are ranked based on their fitness function (Total Error Residual) values.

4.5 EXTENDED KALMAN FILTERING – STAGE 3

In Stage 3, stochastic estimation techniques that use dynamic state space models of the system are provided with the best candidates from the deterministic sampling stage and NSGA – II algorithm for 'Parameter Tuning & System Identification'. These stochastic estimators improve parameter estimates by accounting for sensor and process noise.

4.5.1 Introduction

Three major shortcomings of deterministic models and control theories that prevent direct implementation for parameter estimation are [53]:

1. Mathematical models involve approximations. Capture of complete dynamics of a system using a model is imperfect.
2. A system is driven by a deterministic control input and other disturbances that cannot be modeled deterministically.
3. Sensors are noisy and random noise cannot be modeled deterministically.

Three fundamental estimation problems are depicted in Figure 4.7. For 'Filtering', the time of an estimate coincides with the last measurement point. For 'Smoothing', the time of an estimate falls within the span of available measurement data. For 'Prediction', the

time of estimate occurs after the last available measurement. Fault Diagnosis detects and isolates faults in terms of system states and parameters. In model based diagnostics, the problem of interest is 'Filtering'.

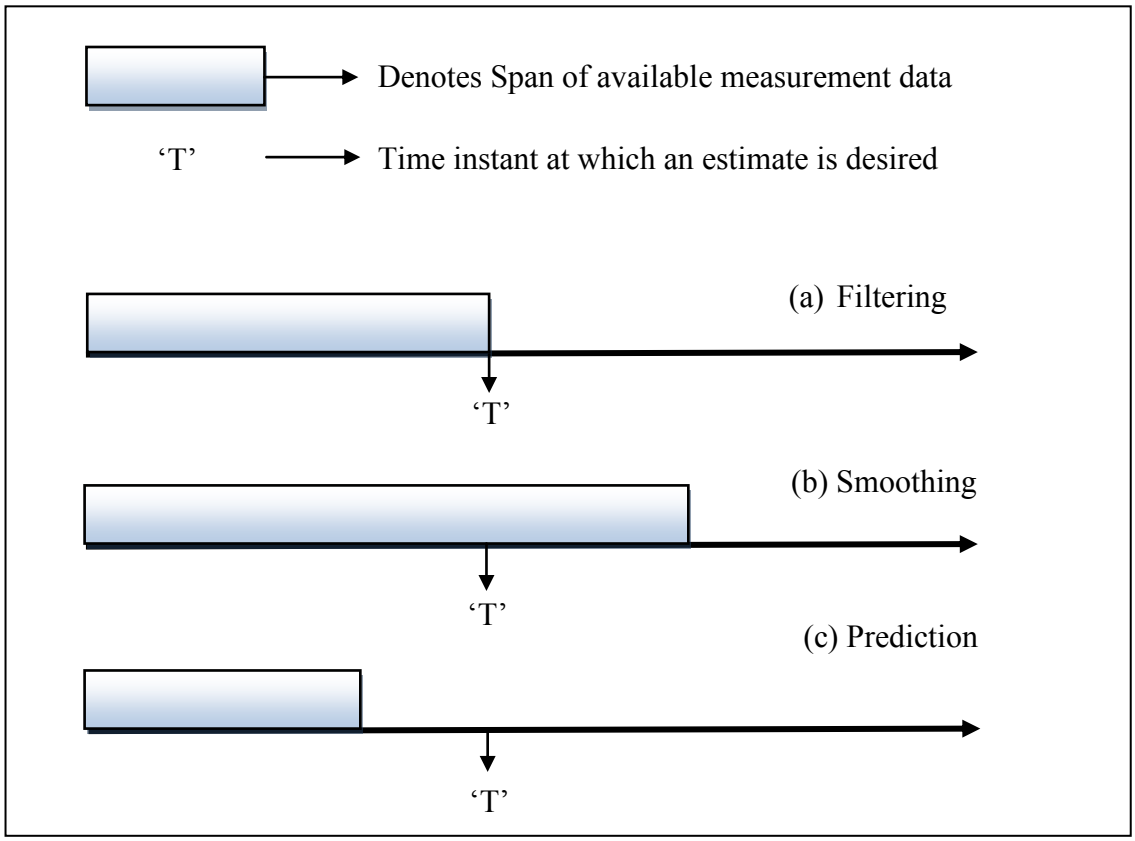

Figure 4.7 Three types of estimation problems [54]

An optimal estimator is an algorithm to process measurements to deduce a minimum error estimate of the state of the system utilizing [54]:

1. Knowledge of system and measurement dynamics
2. Assumed statistics of system noises and measurement errors
3. Initial condition information

4.5.2 Kalman Filter

The "Kalman filter" [55] is a recursive data processing algorithm that obtains an optimal estimate with respect to some criterion. Other data processing techniques such as Weiner filter required all previous data to be stored and re-processed for every single measurement. The recursive Kalman filter enables sequential processing of measurement data, rendering this filter elegant and practical.

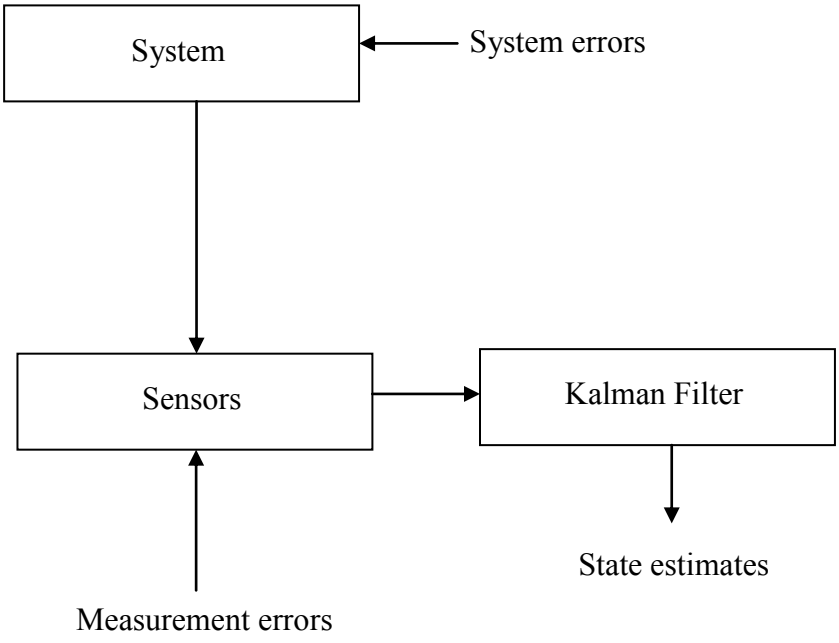

Figure 4.8 Schematic sketch of System, Measurement & Estimator

4.5.2.1 Concept of the Kalman Filter

Consider estimation of a scalar constant x, based on $'k'$ noise - corrupted measurements $Z_i = x + v_i$ where $i = 1,2,3, \ldots . k$. The measurement noise $'v_i'$ is assumed to be white noise.

An unbiased estimate $\hat{x}_k$ is based on an average over all the measurements Z_i

$$\hat{x}_k = \frac{1}{k} \sum_{i=1}^{k} z_i \tag{4.12}$$

When an additional measurement becomes available, the new unbiased estimate is

$$\hat{x}_{k+1} = \frac{1}{k+1} \sum_{i=1}^{k+1} z_i \tag{4.13}$$

The expression can be manipulated as:

$$\hat{x}_{k+1} = \frac{k}{k+1}\, \hat{x}_k + \frac{1}{k+1}(z_{k+1}) \tag{4.14}$$

$$\hat{x}_{k+1} = \hat{x}_k + \frac{1}{k+1}(z_{k+1} - \hat{x}_k)$$

The new estimate $\hat{x}_{k+1}$ is given by the prior estimate $\hat{x}_k$ plus a weighted difference $z_{k+1} - \hat{x}_k$ between the new measurement z_{k+1} and the prior estimate, referred to as the measurement residual.

Extending to vector form, the Kalman filter estimates the state vector $\bar{x} \in R^n$ of a discrete time controlled process governed by the linear stochastic equation [56] :

$$\bar{x}_k = A\bar{x}_{k-1} + B\bar{u}_{k-1} + \bar{w}_{k-1} \tag{4.15}$$

with measurements $Z \in R^n$, given by

$$\bar{Z}_k = H\bar{x}_k + \bar{v}_k \tag{4.16}$$

Random variables $\bar{w}_k$ and $\bar{v}_k$ represent the process and measurement noises respectively. They are assumed to be independent of each other with white power spectra, and normal probability distributions

$$p(w) \sim N(0, Q) \tag{4.17}$$
$$p(v) \sim N(0, R)$$

Here $'Q'$ and $'R'$ are the process and measurement covariance matrices, respectively.

In equation 4.15, the matrix $'A'$ relates the state $\bar{x}_k$ at the previous time step $k-1$ to the state at current time step k, in the absence of process noise. Matrix B relates the control input $\bar{u}$ to the state $\bar{x}$. The matrix H relates the measurement $\bar{Z}_k$ to the state $\bar{x}_k$. The structure of the filter can be classified into two broad categories – (1) Time update and (2) Measurement update. The time update utilizes the deterministic system dynamics to predict the states and covariance of the state estimates at the next time step. The measurement update incorporates an innovation (or) correction factor to correct the $a\ priori\ estimate$ once a measurement at that time step is available. Over time, the real measurement and process noises are slowly included into the model enabling more accurate predictions about the system. Figure 4.9 depicts the control flow in a Kalman filter.

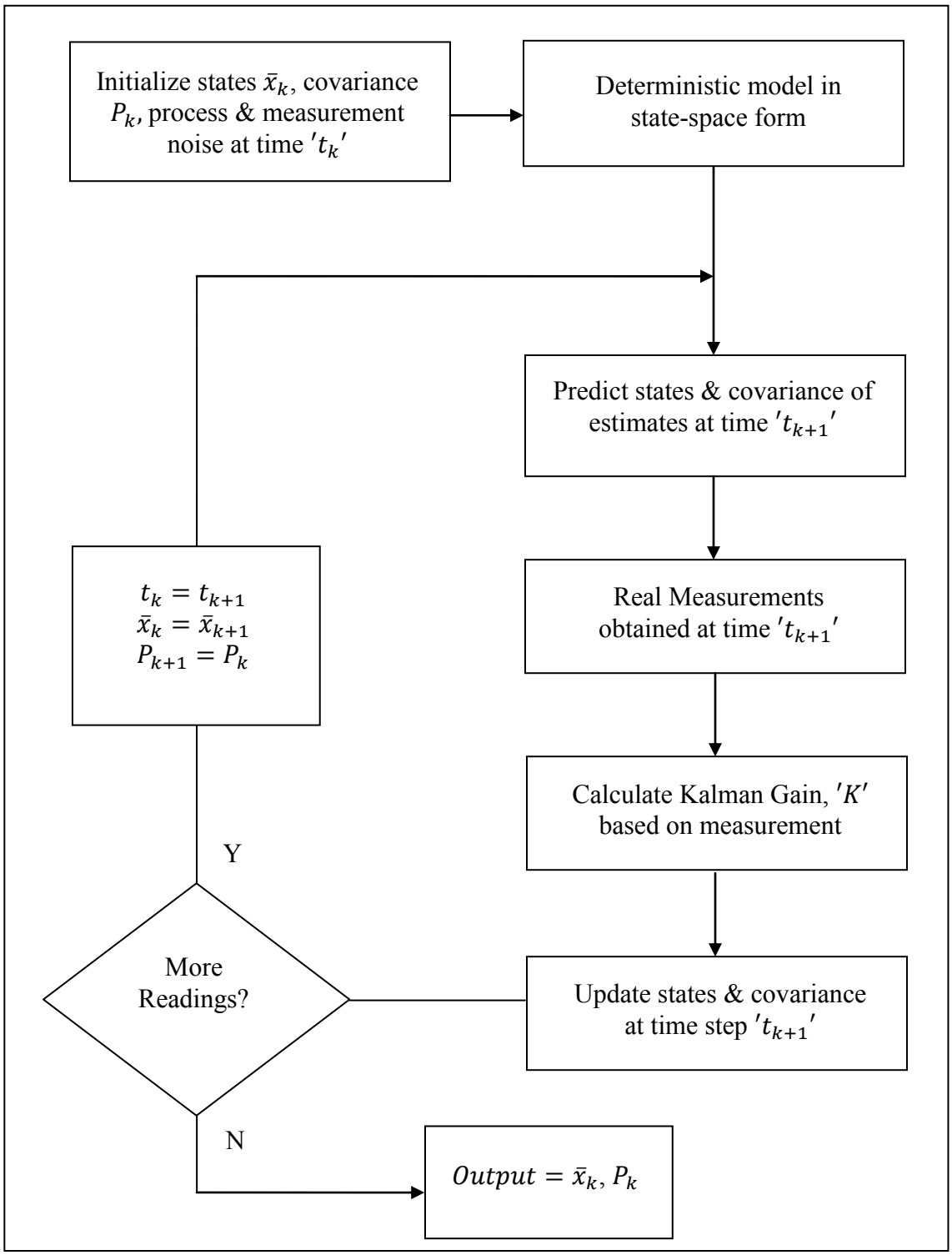

Figure 4.9 Schematic sketch of control flow in a Kalman Filter Algorithm

4.5.2.2 Mathematical Formulation

First, the Kalman Filter assigns an $a\ priori$ state estimate $\bar{x}_k^- \in R^n$ at time step $'k'$, where the minus sign indicates an $a\ priori$ estimate of the state before an actual measurement at that time step is obtained. Let $\hat{x}_k \in R^n$ be the $a\ posteriori$ state estimate done after the measurement at time step $'k'$. The $a\ priori$ and $a\ posteriori$ state estimate errors are defined by the equation:

$$\begin{aligned} \bar{e}_k^- &= \bar{x}_k - \bar{x}_k^- \\ \bar{e}_k &= \bar{x}_k - \hat{x}_k \end{aligned} \tag{4.18}$$

The $a\ priori$ state estimate error covariance is

$$\bar{P}_k^- = E\left[\bar{e}_k^- \bar{e}_k^{-T}\right] \tag{4.19}$$

and the $a\ posteriori$ state estimate error covariance is

$$\bar{P}_k = E\left[\bar{e}_k \ \bar{e}_k^{\ T}\right] \tag{4.20}$$

Similar to the previous simple example, the Kalman filter computes the $a\ posteriori$ state estimate as a linear combination of $a\ priori$ state estimate and a weighted difference of the measurement residual or 'innovation'.

$$\hat{x}_k = \bar{x}_k^- + K(\bar{Z}_k - H\bar{x}_k^-) \tag{4.21}$$

The residual $\bar{Z}_k - H\bar{x}_k^-$ measures the discrepancy between the actual measurement at a particular time step, and the predicted measurement computed via the measurement function model. As time progresses, the model updates and residuals consistently reduce.

Matrix $'K'$ - the 'Kalman Gain' or 'blending factor' minimizes the $a\ posteriori$ error covariance. The minimization can be performed by substituting (4.20) in equation (4.17), followed by expectations defined in (4.19). Taking derivative of the trace, setting it to zero for a minimum, and solving gives:

$$K_k = \bar{P}_k^{-} H^T (H\bar{P}_k^{-} H^T + R)^{-1} \tag{4.22}$$

or

$$K_k = \frac{\bar{P}_k^{-} H^T}{(H\bar{P}_k^{-} H^T + R)} \tag{4.23}$$

As the measurement noise covariance $'R'$ reduces and approaches zero, the residuals $H\bar{P}_k^{-} H^T$ become prominent, because of trust gained about incoming measurements. In short,

$$\lim_{R \to 0} K_k = H^{-1} \tag{4.24}$$

As the $a\ priori$ state estimate covariance $\bar{P}_k^{-}$ approaches zero, the model is sufficiently updated to make accurate predictions of individual states. The Kalman gain value approaches zero, which causes the algorithm to ignore the residuals. This can be simply represented as

$$\lim_{\bar{P}_k^{-} \to 0} K_k = 0 \tag{4.25}$$

When measurement noise covariance is small, the filter "trusts" $\bar{Z}_k$ more, and ignores measurement model readings. When the $a\ priori$ state estimate covariance is low, the kalman gain shrinks, and more emphasis is given to the measurement prediction $H\bar{x}_k$ term. The equations that govern the Kalman filter are shown in Figure 4.10.

- Time Update :
  - $\bar{x}_k^- = A\hat{x}_{k-1} + B\hat{u}_{k-1}$
  - $\bar{P}_k^- = A\hat{P}_{k-1}A^T + Q$
- Measurement Update :
  - $K_k = \bar{P}_k^- H^T (H\bar{P}_k^- H^T + R)^{-1}$
  - $\hat{x}_k = \bar{x}_k^- + K(\bar{Z}_k - H\bar{x}_k^-)$
  - $P_k = (I - K_k H)\bar{P}_k^-$

Figure 4.10 Kalman Filter Equations

4.5.2.3 Filter Tuning

By observing the noise in the sensor measurements, the measurement noise covariance $'R'$ can be computed off-line. More challenging is the process noise covariance matrix $'Q'$ because the process is not directly observable in a physical sense. Tuning parameters in the filter are the $'Q'$ and $'R'$ matrices. Sometimes, a relatively poor process model can produce acceptable results if enough process noise is injected into the model, and if appropriate values for $'Q'$ and $'R'$ are selected. If the values for both covariance matrices are chosen off-line and are assumed constant, then the filter can stabilize quickly and remain constant, but may not be correct.

In real systems, often fixed values for $'Q'$ and $'R'$ do not result in accurate estimates due to variations in sensor measurements and the process model. For example, while trying to measure the distance of an object using SONAR, the sensor might render

measurements with less noise for closer objects compared to objects far away. A very specific or rare condition might not be captured adequately by the process model, resulting in sudden variation of the system states and parameters. The process uncertainty injected through the value of $'Q'$ initially might not be enough to capture this sudden variation. Further complications arise when the system is non-linear. Constructing an analogy to an optimization search technique, the $'Q'$ and $'R'$ matrices can be equated to search regions in which the optimization algorithm should operate. The filter algorithm searches for values within these search regions that minimize the state estimate covariance. The search is directed via gradient based information available in the deterministic model embedded in the filter. The search regions are clouds of uncertainty around the actual values of the states. As time progresses and the number of incoming measurements increases, the algorithm tries to shrink this uncertainty, thereby giving accurate estimates for the states. Needed are mechanisms which vary the process and measurement covariances in a dynamic manner, to improve the filter output. State Noise Compensation (SNC) and Dynamic Model Compensation (DMC) are popular techniques to dynamically vary the process noise covariance and capture system variations more precisely.

4.5.3 Extended Kalman Filter

The Kalman filter was designed to estimate the state $\bar{x} \in R^n$ of a discrete time controlled process governed by a linear stochastic equation. The "extended" Kalman Filter was designed for processes governed by a non-linear relationship

$$\tilde{x}_k = f(\bar{x}_{k-1}, \bar{u}_{k-1}, \bar{w}_{k-1}) \tag{4.26}$$

with measurement

$$\tilde{Z}_k = h(\bar{x}_k, \bar{v}_k) \tag{4.27}$$

The non-linear function $'f'$ relates the state at time step $'k'$ to the previous time step $'k-1'$. Here $\bar{u}_{k-1}$ is a deterministic input and $\bar{w}_{k-1}$ is an unknown process noise. Non-linear mapping $'h'$ relates the measurements $\tilde{Z}_k$ to the states $\tilde{x}_k$ at any time step $'k'$. The statistical properties of the process noise $\bar{w}_k$ and measurement noise $\bar{v}_k$ are similar to the linear case.

To estimate the states of the non-linear system, the extended Kalman Filter linearizes the non-linear process equation as

$$\bar{x}_k \approx \tilde{x}_k + A\left(\bar{x}_{k-1} - \hat{x}_{k-1}\right) + W(\bar{w}_{k-1}) \tag{4.28}$$

Similarly, the measurement equation can be linearized as

$$\bar{Z}_k \approx \tilde{Z}_k + H(\bar{x}_k - \tilde{x}_k) + V(\bar{v}_k) \tag{4.29}$$

Here,

- $\bar{x}_k$ and $\bar{Z}_k$ are the actual state & measurement vectors
- $\tilde{x}_k$ and $\tilde{Z}_k$ are the expected state & measurement vectors
- $\hat{x}_{k-1}$ is the $a\ posteriori$ state estimate at time step $'k-1'$
- A is the Jacobian of function $'f'$ with respect to state $\bar{x}$
  - $A_{[i,j]} = \frac{\partial f_{[i]}}{\partial x_{[j]}}$
- W is the Jacobian of function $'f'$ with respect to $\bar{w}$
  - $W_{[i,j]} = \frac{\partial f_{[i]}}{\partial w_{[j]}}$
- H is the Jacobian of function $'h'$ with respect to state $\bar{x}$
  - $H_{[i,j]} = \frac{\partial h_{[i]}}{\partial x_{[j]}}$
- V is the Jacobian of function $'h'$ with respect to noise $\bar{v}$
  - $V_{[i,j]} = \frac{\partial h_{[i]}}{\partial v_{[j]}}$

Since the Jacobian matrices evolve with time, the Jacobians must be computed at each time step. The prediction error cannot be calculated in terms of the states because the actual state $\bar{x}_k$ at time instant $'k'$ is not known. On the other hand, the actual measurement $\bar{Z}_k$ at the same time instant $'k'$ is available through sensor readings. Therefore the state prediction error and measurement prediction errors

$$\tilde{e}_{x_k} = \bar{x}_k - \tilde{x}_k \qquad (4.30)$$
$$\tilde{e}_{z_k} = \bar{z}_k - \tilde{z}_k$$

can be manipulated as

$$\tilde{e}_{x_k} \approx A\left(\bar{x}_{k-1} - \hat{x}_{k-1}\right) + \in_k \qquad (4.31)$$
$$\tilde{e}_{z_k} \approx H\tilde{e}_{x_k} + \eta_k$$

where $\in_k$ and η_k are new independent random variables having zero mean and covariance matrices WQW^T and VRV^T. The complete extended Kalman Filter equations are given in Figure 4.11.

- Time Update :
  - $\hat{x}_k^- = f(\hat{x}_{k-1}, \hat{u}_{k-1}, \overline{w}_{k-1})$
  - $\bar{P}_k^- = A_k P_{k-1} A_k^T + W_k Q_{k-1} W_k^T$
- Measurement Update :
  - $K_k = \bar{P}_k^- H_k^T \left(H_k \bar{P}_k^- H_k^T + V_k R_k V_k^T\right)^{-1}$
  - $\hat{x}_k = \bar{x}_k^- + K_k(\bar{Z}_k - h(\bar{x}_k^-, \bar{v}_k))$
  - $P_k = (I - K_k H)\bar{P}_k^-$

Figure 4.11 Extended Kalman Filter equations

In conclusion, a stochastic estimator like the Kalman filter approximates the state distribution by a Gaussian Random Variable, which is then propagated analytically through the dynamic model embedded in the filter. For a linear system, this propagation is accurate and the filter produces the optimal state estimate with least estimate covariance. For the extended Kalman Filter, though the state distribution is assumed Gaussian, the propagation is based purely on a first order linearization of the non-linear system. This fundamental flaw might result in large errors in the true mean and covariance of the $a\ posteriori$ transformed Gaussian variable [57], resulting in sub-

optimal performance, and filter divergence due to accumulation of errors over a period of time. If there is not a one-to-one mapping between the measurement and the state, i.e., if the process is unobservable, the extended Kalman filter will quickly diverge, and there is no guarantee of optimality in the estimation of states [56]. This underscores why an observability analysis prior to parameter estimation is critical. Observability analysis indicates which combination of measurements can render acceptable parameter estimates.

The module could operate as a hybrid "batch plus sequential" estimator. Stage 1 (Hammersley Sampling) and Stage 2 (NSGA-II) operate on a set of measurement data extracted over a fixed period of time. The output parameter values would be given to the Extended Kalman filter stage which constantly updates system states and parameters based on incoming measurements. This control flow is applicable only when the module is first installed. For the next cycle, a moving time window could automatically extract a new measurement data set for the Hammersley sampling and NSGA-II algorithms; the extended Kalman filter would run in parallel in the background. Parameter values obtained using the new batch estimation process could be compared with the current parameter values estimated by the Extended Kalman filter. Future work could implement this idea and formulate a procedure to make decisions based on outputs of batch and sequential estimators.

Chapter 5: Implementation & Results

The "Parameter Tuning" module was implemented and tested for two different cases:

1. Experimental study on a DC Motor
2. Simulation based study on a 3-phase induction motor

5.1 EXPERIMENTAL STUDY ON A DC MOTOR

A DC motor was interfaced with a PC using LabVIEW data acquisition module and measurements were then taken.

5.1.1 State Space modeling of a DC Motor

The DC motor model relates the motor's input voltage to output velocity. The DC motor circuit is represented by the schematic sketch in Figure 5.1.

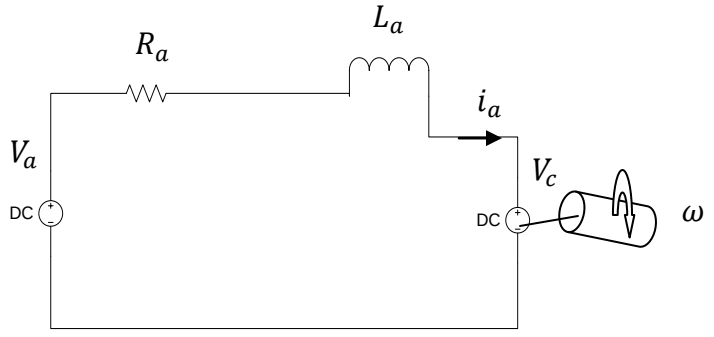

Figure 5.1 Electrical representation of a DC motor

Applying Kirchhoff's voltage law to figure 5.1 gives

$$V_a - i_a R_a - L_a \frac{d}{dt} i_a - K_b \omega_a = 0 \tag{5.1}$$

where,

$'i_a'$ is the armature current

$'L_a'$ is the inductance of the armature coil

$'K_b'$ is the back-emf constant of the motor

$'\omega_a'$ is the angular velocity of the motor

Euler's law sums the torques on the rotor giving

$$i_a k_t - J\frac{d}{dt}\omega_a - B\omega_a - \tau_l = 0 \tag{5.2}$$

where,

$'k_t'$ is the motor torque constant and is same as the back-emf constant

$'J'$ is the inertia of the rotor

$'B'$ is the damping co-efficient associated with the mechanical rotation

$'\tau_l'$ is the load torque

The differential equations (5.1) and (5.2) for the armature current and the angular velocity of the motor can be written as

$$\frac{d}{dt}i_a = -\frac{R_a}{L_a}i_a - \frac{K_b}{L_a}\omega_a + \frac{V_a}{L_a} \tag{5.3}$$

$$\frac{d}{dt}\omega_a = \frac{k_t}{J}i_a - \frac{B}{J}\omega_a - \frac{\tau_l}{J} \tag{5.4}$$

5.1.2 Characterization of the DC motor

To characterize the DC motor and obtain values for the various parameters of the motor, the inductance term $'L_a \frac{d}{dt} i_a'$ of the motor was initially neglected under the assumption of steady state. The back-emf of the motor was measured and the motor constant $'K_b'$ calculated as

$$K_b = \frac{V_{backemf}}{\omega_{ss}} \tag{5.5}$$

where $'\omega'_{ss}$ is the steady state velocity of the motor. From the voltage balance equation, the resistance of the motor can be calculated as

$$R_a = \frac{V_a - K_b \omega_{ss}}{i_{ss}} \tag{5.6}$$

where $'i'_{ss}$ is the measured steady state current. The rotor was removed from the motor and weighed. From the weight and geometry of the rotor, the inertia $'J'$ of the rotor was estimated.

Similarly, at steady state, the acceleration of the rotor is zero, and the torque equation (5.2) reduces to

$$i_a k_t - B\omega_a - \tau_l = 0 \tag{5.7}$$

The velocity $'\omega'_a$ of the motor was varied and the corresponding $'i'_a$ values were measured. Since the motor constant $'k'_t = 'k'_b$, a linear fit between velocity and motor current rendered values of damping coefficient $'B'$ and load torque $'\tau_l'$. Values of various measured parameters are tabulated in Table 5.1.

Table 5.1: Parameters of the DC motor

Parameters	Description	Value
J	Inertia (kg-m^2)	1.8201E-7
B	Damping Coefficient (Ns/m)	0.000000155277950
R_a	Resistance of rotor (Ω)	2.312976182
k_t	Motor Constant (Nm/A)	0.00362736
τ_l	Load Torque (Nm)	0.000244148580241

5.1.3 Results for DC motor

The velocity of the motor was varied by suitably adjusting the input voltage and current. The velocity of the motor $'\omega_a'$ was tracked to estimate the resistance of the rotor $'R_a'$. The Total Error Residual function (cost function) is

$$Total\ Error\ Residual = w_1 * r_1 \tag{5.8}$$

where weight $'w_1'$ was chosen as 1 and now the residual $'r_1'$ is to be computed. To calculate the residual with respect to an output tracked, the state space model is utilized.

Figure 5.2 shows the manner in which input voltage was varied. Values for parameters were set in accordance to Table 5.1 except for the resistance of the rotor (R_a) which is to be estimated.

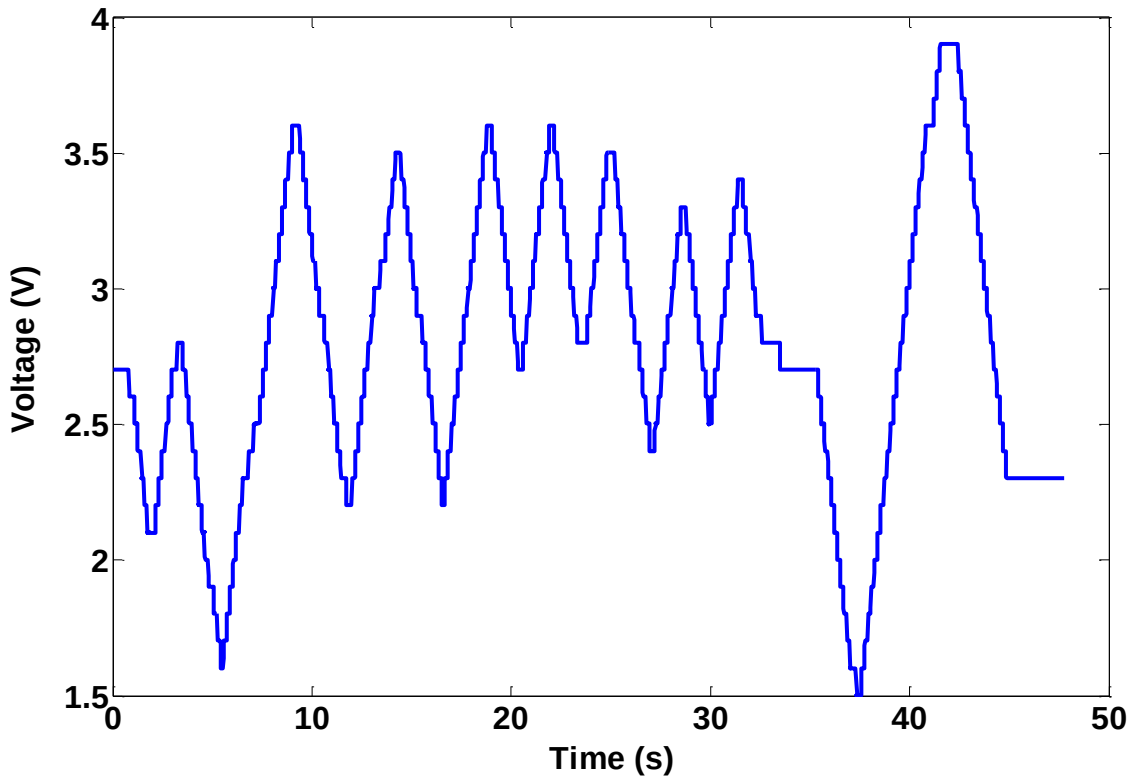

Figure 5.2 Input Voltage supplied to motor

The diagnostic module initially queries for the number of parameters to be estimated. Here is only one unknown parameter - $'R_a'$. The next query specifies the nominal bounds for the parameters. In this case, 1Ω< R_a <10 Ω. The next query sets the number of sample points. Since the model is simple with only one parameter to estimate, the number of sample points is set to 5. The output velocity was simulated for each of the

parameter sample points generated by the Hammersley sequence using a fixed step 'Runge-Kutta-4' solver. The initial values for the states was set to 0 and the time span for the solver was set to [0 50] seconds as per measurement data available.

To compute the residual at a sample point, the simulated signal must be correlated time-wise to the measured signal, since time lags can increase falsely the residuals. Once the signals have been correlated, absolute values of differences are taken at each time step to obtain the residual (noise) signal. This noise signal is accumulated over time to obtain a scalar value which is basically the integral of the noise curve. Figure 5.3 shows the noise signal for one of the resistance values (sample point).

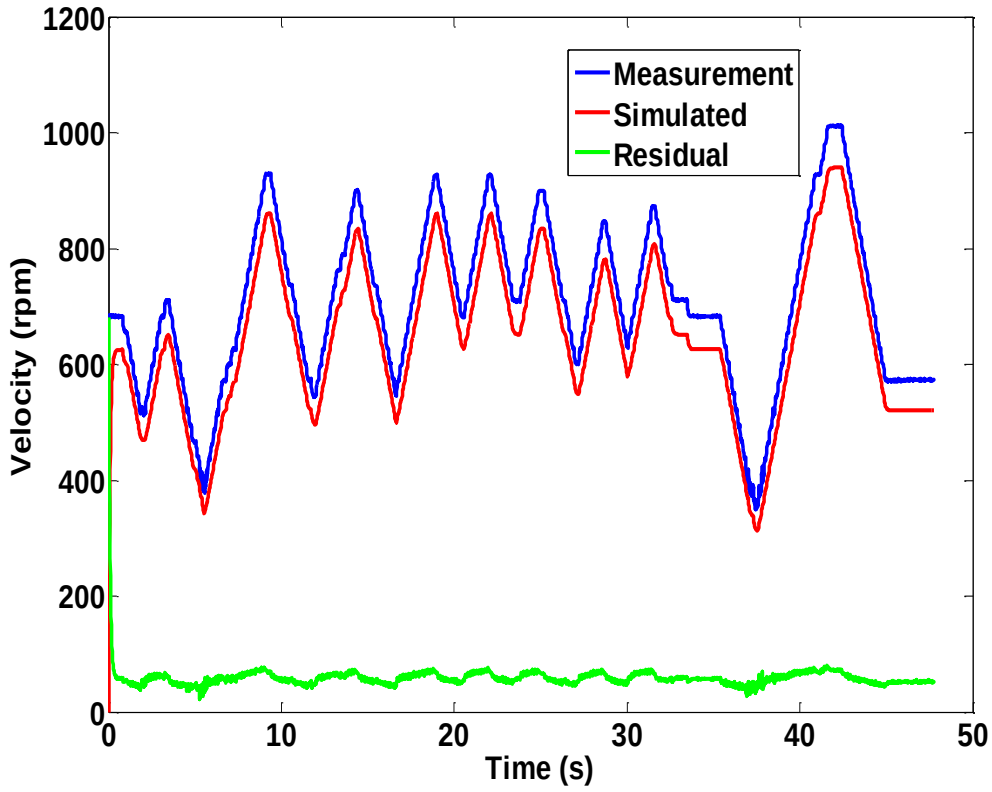

Figure 5.3 Noise signal between measured and simulated velocities

The parameter tuning algorithm is a multi-step process, wherein parameters evolve over a fixed number of generations (in this case, 5). The objective is to obtain a ball-park of the actual operating conditions of the system. Each set of five points generated using the Hammersley sequence constitutes a "progeny". The five sample points are now ranked based on their error residuals. The point with smallest error residual is passed into the generateArea ( ) function, which computes new bounds for each parameter, thereby defining the search region for next progenies. New upper and lower bounds were assigned to each parameter as $P_i(1+\Delta)$ and $P_i(1-\Delta)$, where $\Delta = 0.5$.

Figure 5.4 shows the evolution of the algorithm over different progenies. Based on the best point of each set of progenies, a new search domain is defined for the next set of sample points. A decrease in the search domain for the parameter is apparent with the 'global minimum' always hovering around 2.2 ohm. From an uncertainty of 1-10 ohm (X-axis in figure 5.4a), the algorithm has narrowed the search region to 1-4 ohm (X-axis in figure 5.4e), with just 5 sample points. Also, significant decrease in the cost function value is noted over progenies with first generation progenies having values around 16 x 10^6 (Y-axis in figure 5.4a) and the last generation progenies having values within 3.5 x 10^6 (Y-axis in figure 5.4e).

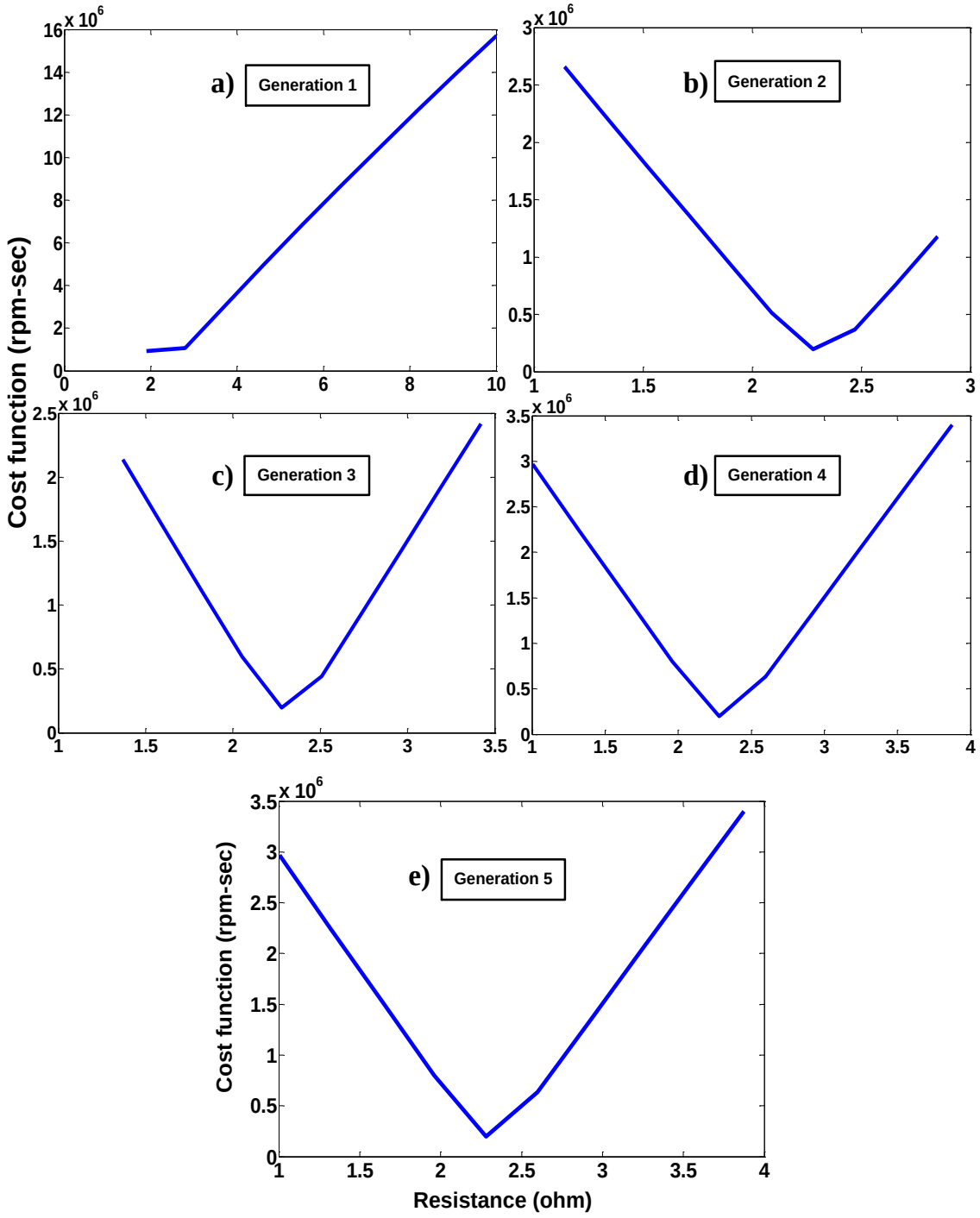

Figure 5.4 Uncertainty reductions across different generations

Finally the cost function of all the points across various generations ranging from the initial bounds 1-10 are plotted and shown in figure 5.5. The plot displays the cost function over the entire span. The cost function minimum suggests that the resistance $'R_a'$ is around 2.2 ohm which agrees with the actual measured resistance - 2.3 ohm. It also emphasizes on the fact that though the models are not complex enough to capture every phenomenon occurring inside the motor, simpler models can work well as long as variation of the parameter of interest with respect to the output tracked is captured reasonably well.

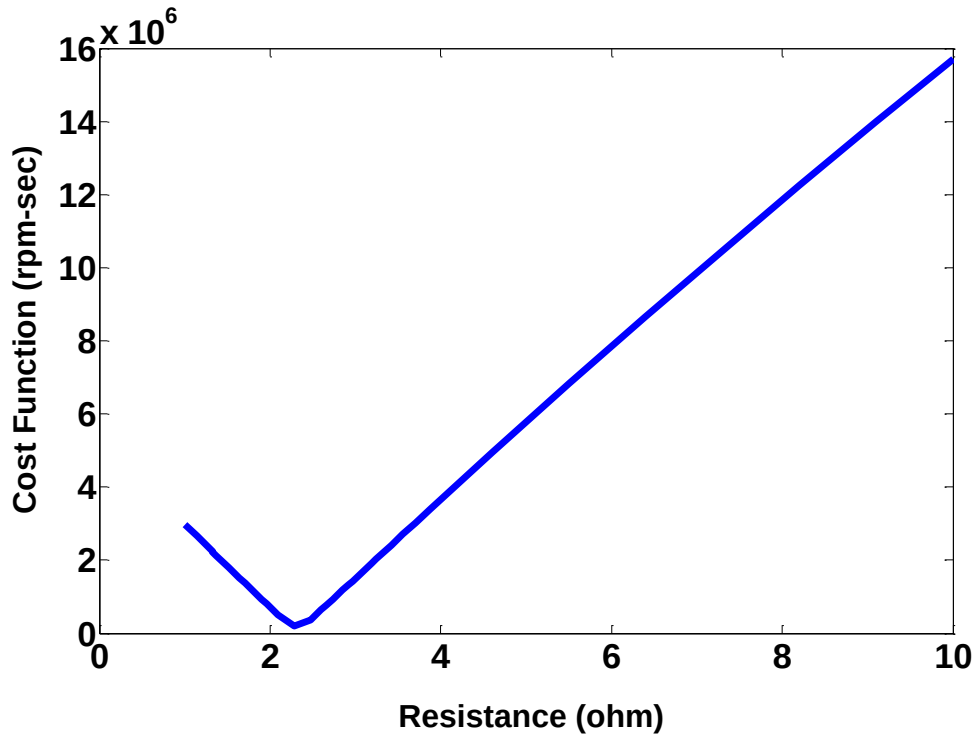

Figure 5.5 Variation of Total Error Residual function across all generations

The module was tested on a different velocity measurement data in figure 5.6. The cost function Vs R_a for all the sample points across various generations is shown in figure 5.7. From the plot, the global minimum of the Total Error Residual function again lies around 2.3 ohm.

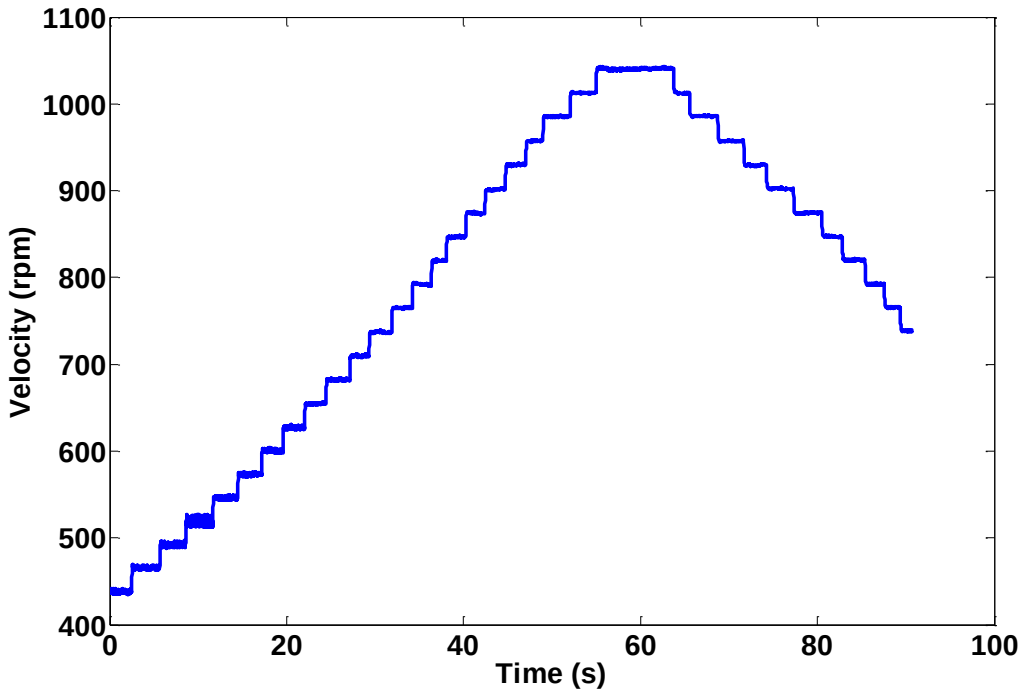

Figure 5.7 Measured output velocity (data set -2) of the motor

Only a few parameters change and evolve (degrade) over time and usage. In the DC motor, inertia of the rotor and the motor constant vary only slightly for almost the entire life of the motor. For each fault, only certain parameters need to be tracked. This reduced number of parameters to tune focuses the tuning process to a smaller sub-space. This subspace will have lower dimensionality.

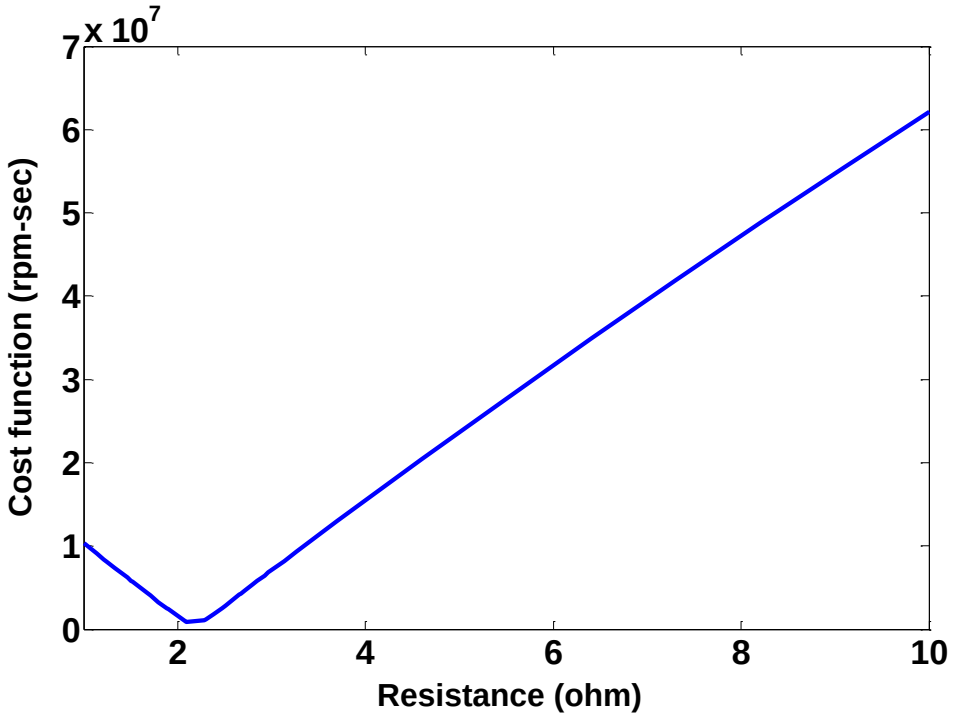

Figure 5.8 Variation of Total Error Residual function for II test data

The lower dimension sub-space increases the ability to track the operation condition of the system. Table 5.2 lists important functions used in this module and their respective input and output parameters.

5.1.4 Algorithm and Computational specifications

Tuning module specifications and computational resources used included:

- Number of processor cores – 1
- Clock Speed of processor – 2.99 GHz
- Algorithm run time – 117.291894 seconds. (MATLAB tic-toc function)
- Number of generations for algorithm – 5

Table 5.2: Functions Used

Function	Input Parameters	Output
Hammersley	Bounds, number of samples	Hammersley sequence sample points
Rk4_fixed	Model, Initial State vector, Time span & number of steps	Time response of the system
Error_function1	Sample Point	Residual area
xcorAlign	Signals to be correlated	Correlated signals
generateArea	Best point in a generation	New search region for the next generation

5.2 SIMULATION BASED STUDY ON A 3-PHASE INDUCTION MOTOR

A simulation based study of the parameter tuning module used the induction motor model in chapter 3. To assess the performance of the module, a stator coil fault was chosen. Stator faults can be caused by a cracked component or loose connection in the stator circuit, and can be emulated by adding a resistance in series to the stator coil.

5.2.1 Characterization of the 3-phase induction motor

Values of parameters used in the induction motor model are based on the experimental study by Choi [58]. The system parameter values used for the simulation are shown in table 5.3.

Table 5.3: Parameter Values used in state space model for induction motor

Parameter	Description	Healthy Value
R_s	Stator Coil Resistances (Ω)	2.1
R_r	Rotor Bar Resistance (Ω)	0.8663
L_s	Stator Inductances (H)	1.02938
L_r	Rotor Inductances (H)	0.9834
L_m	Mutual Inductance (H)	1.00130
V_{peak}	Peak Voltage (V)	230
f	Input Frequency (Hz)	60
R_{br}	Mechanical Friction (N-s/m)	0.0085
R_{sm}	Stator Magnetic loss (Ω)	0.03539
n	Number of rotor bars	5
J	Moment of Inertia (N-m^2)	0.0115
c	Mechanical Friction (Ns/m)	0.0085

5.2.2 Simulation set-up

The induction motor model is highly non-linear. Multiple outputs tracked include:

1. Velocity of the motor - ω
2. Stator currents for three phases - i_a , i_b and i_c
3. Angular displacement of the shaft – θ

With 5 outputs being tracked, the Total Error Residual (TER) function can be formulated as

$$TER = (w_1 * \omega_r) + (w_2 * (i_{ra} + i_{rb} + i_{rc})) + (w_3 * \theta_r) \quad (5.9)$$

where,

w_1, w_2, w_3 - weights for residual functions

ω_r – velocity residual

i_{ra}, i_{rb}, i_{rc} - residuals for 3-phase stator currents

θ_r - residual of shaft angular displacement

The minimization problem

$$Minimize_{R_{sa}\, R_{sb}\, R_{sc}} [Total\ Error\ Residual] \quad (5.10)$$

minimizes the total error residual function by appropriately varying the individual stator resistances across the 3-phases - R_{sa}, R_{sb} and R_{sc}.

To test the parameter tuning module, “measurements” will arise from a motor model that emulates a real motor. This model includes faults, and is distinct from the process model used for diagnostics. The “measurement” model substitutes for the real

machine and facilitates convenient testing of the parameter tuning module. The cases emulated by the measurement model are:

1. Resistance of one phase doubled
2. Two of the phase resistances doubled.
3. All three phase resistances doubled
4. Two phase resistances doubled, the third phase resistance tripled.

5.2.3 Case 1 – Resistance of one phase doubled

The measurement model doubled the stator resistance across phase $'a'$ (2.1 x 2 = 4.2 ohm). Since three stator resistances will be tuned, the number of parameters is set to 3. Since the model is highly non-linear, 150 sample points were chosen for the deterministic algorithm. The algorithm was parallelized on 8 processor cores. The next step bounds the stator resistances. The lower bound for each resistance was set to 1.9 ohm (marginally close to the healthy value of 2.1 ohm) and upper bound was set to 8.4 ohm – (4 times healthy value). The 5 points in the parameter space with smallest error residual cost function as discovered by the deterministic sampling algorithm are shown in Table 5.4 along with residual (cost function) values.

Table 5.4: Top 5 possible points from deterministic sampler – case 1

Case	R_{sa} (Ω)	R_{sb} (Ω)	R_{sc} (Ω)	Cost function
A	3.890304	2.2861875	2.1206564	3.91830369
B	1.960450	3.1040742	3.2039791	5.11292705
C	2.281351	3.7561113	2.2679930	5.43903848
D	4.132800	2.2790588	1.8227404	5.83622560
E	2.167200	3.0065185	3.0791495	6.31642304

Cases A, C and D suggest that one resistance has almost doubled (4.2 ohm), while the other two resistances are close to the original value (2.1 ohm). The cost function values support this proposition. These five top points across all stages of the deterministic sampler were passed to 'Stage 2 – Random Sampling (NSGA-II)'. New regions were defined around these "hot spots" in the parameter space using a ratio $\Delta = 0.35$. New upper and lower bounds were assigned to each parameter as $P_i(1 + \Delta)$ and $P_i(1 - \Delta)$. Randomness in the search, introduced through the NSGA-II algorithm, abets locating the global minimum. The genetic algorithm uses 5 processor cores to parallelize the search over 5 different regions. The genetic algorithm population was 50 and the number of generations was 3. The output of the NSGA-II algorithm was written into a text file "solution (core number).txt" in ASCII format. A total of 3 x 50 = 150 possible candidates were produced with corresponding error residual values. The output candidates from NSGA-II were imported into a matrix with the top 5 candidates from the deterministic algorithm. All candidates were ranked based on error residual values. This step ensures

no deterministic algorithm candidate, possibly better than the genetic algorithm candidates was discarded. The top 5 operating point candidates obtained after the genetic algorithm stage are shown in Table 5.5.

Table 5.5: Top 5 possible points after NSGA module – case 1

R_{sa} (Ω)	R_{sb} (Ω)	R_{sc} (Ω)	Cost function
3.97465	1.94578	2.35294	2.25422
4.00266	1.91960	2.33849	2.63308
4.00499	1.91652	2.33880	2.65184
4.00350	1.91690	2.33872	2.68407
3.96382	1.93025	2.34918	2.91187

All five candidates in Table 5.5 suggest the resistance across phase $'a'$ has almost doubled, with values close to 4.0 ohm (actual value is 4.2 ohm). The resistances across the other phases are close to the actual 2.1 ohm. Comparing the residual values of the sample points obtained from the NSGA-II algorithm with those from the deterministic sampler alone (Table 5.4) suggests that incorporating randomness in the algorithm aids attainment of the global minimum. Figure 5.9 compares the healthy and faulty system responses. Plotted are the 3-phase stator currents, motor velocity and shaft displacement versus time. Column (a) to the left shows "measurements" from the healthy and faulty systems. Column (b) to the right compares model simulations after tuning of parameters to "measurements" from the faulty system. Simulations overlay measurements.

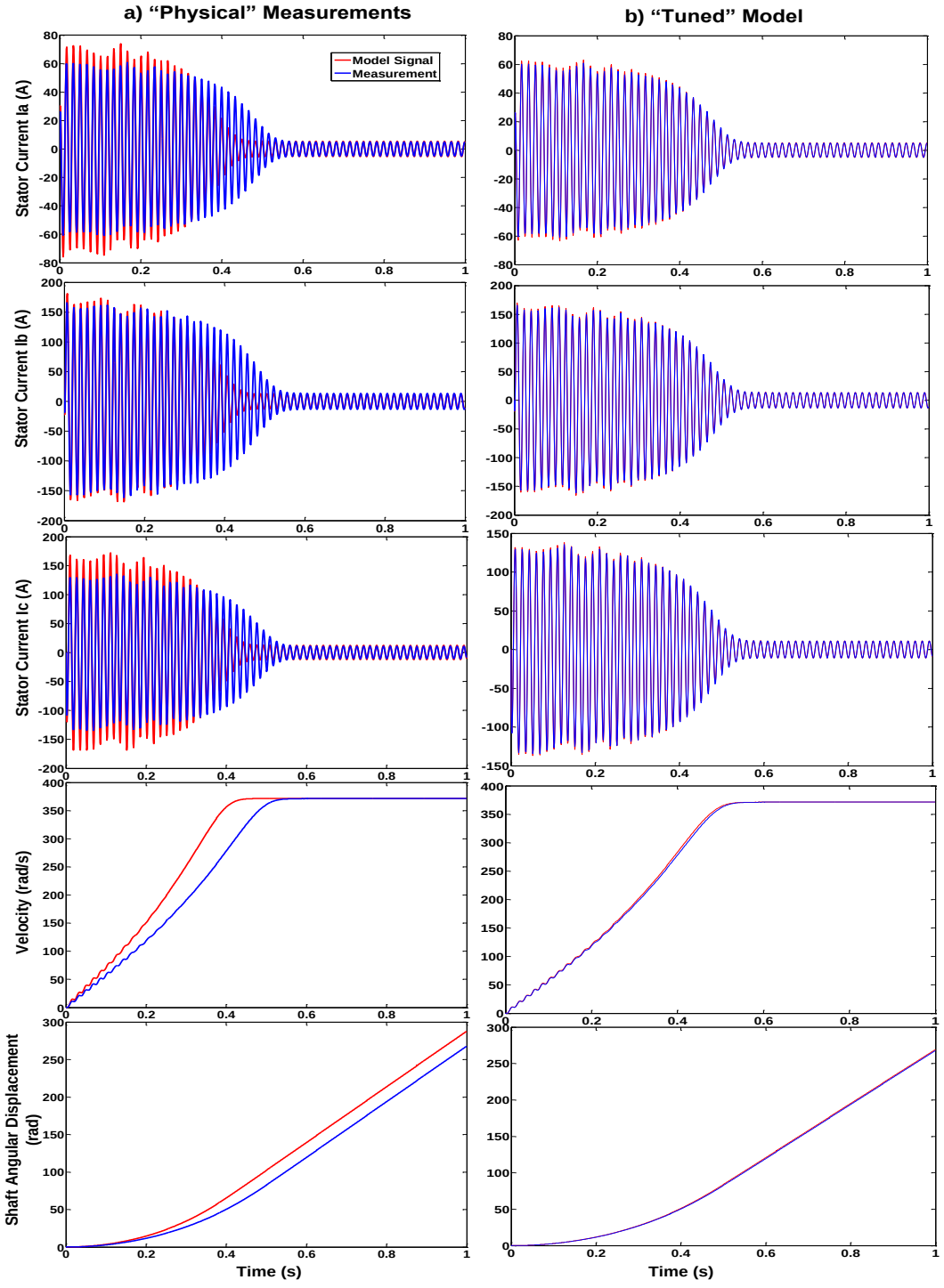

Figure 5.9: (a) Healthy & faulty system "measurements" (b) Comparison of model simulations with tuned parameters to "measurements" – Case 1

5.2.4 Case 2 – Resistance across two phases doubled

In the measurement model, resistance values of phases $'b'$ & $'c'$ were doubled to 4.2 ohm. The procedure followed in the previous section was repeated. The output of the deterministic sampler module is shown in Table 5.6.

Table 5.6: Top 5 possible points post Hammersley sampler – case 2

R_{sa} (Ω)	R_{sb} (Ω)	R_{sc} (Ω)	Cost function
3.3409	3.0771	3.9901	7.6628597
3.4030	3.1492	3.8049	8.0733625
2.7830	4.0765	3.5867	8.7199573
4.4567	2.3275	3.7212	9.0413446
3.4900	3.5496	3.3653	9.2556587

Although the deterministic sampler module was not as proficient as in the previous section, the deterministic sampler's goal is to get within the ball park of the operating region of the system and define boundaries for the genetic algorithm stage. The five points of Table 5.6 were passed to the genetic algorithm stage

Table 5.7: Top 5 possible operating points after NSGA module – case 2

R_{sa} (Ω)	R_{sb} (Ω)	R_{sc} (Ω)	Cost function
2.22960	3.94739	4.39591	4.94034
2.22961	3.94525	4.39598	5.13046
2.22959	3.94726	4.39596	5.19443
2.22951	3.94748	4.39583	5.22857
2.22966	3.94732	4.39598	5.28148

Results from the NSGA-II genetic algorithm stage shown in Table 5.7 suggest a doubling of resistances across phases $'b'$ & $'c'$, in agreement with the measurement model simulated with two resistances doubled. Figure 5.10 plots the healthy model measurements versus faulty system measurements fig. 5.10 (a) and simulation results after tuning fig. 5.10 (b). Similar to fig.5.9, model simulations overlay measurements.

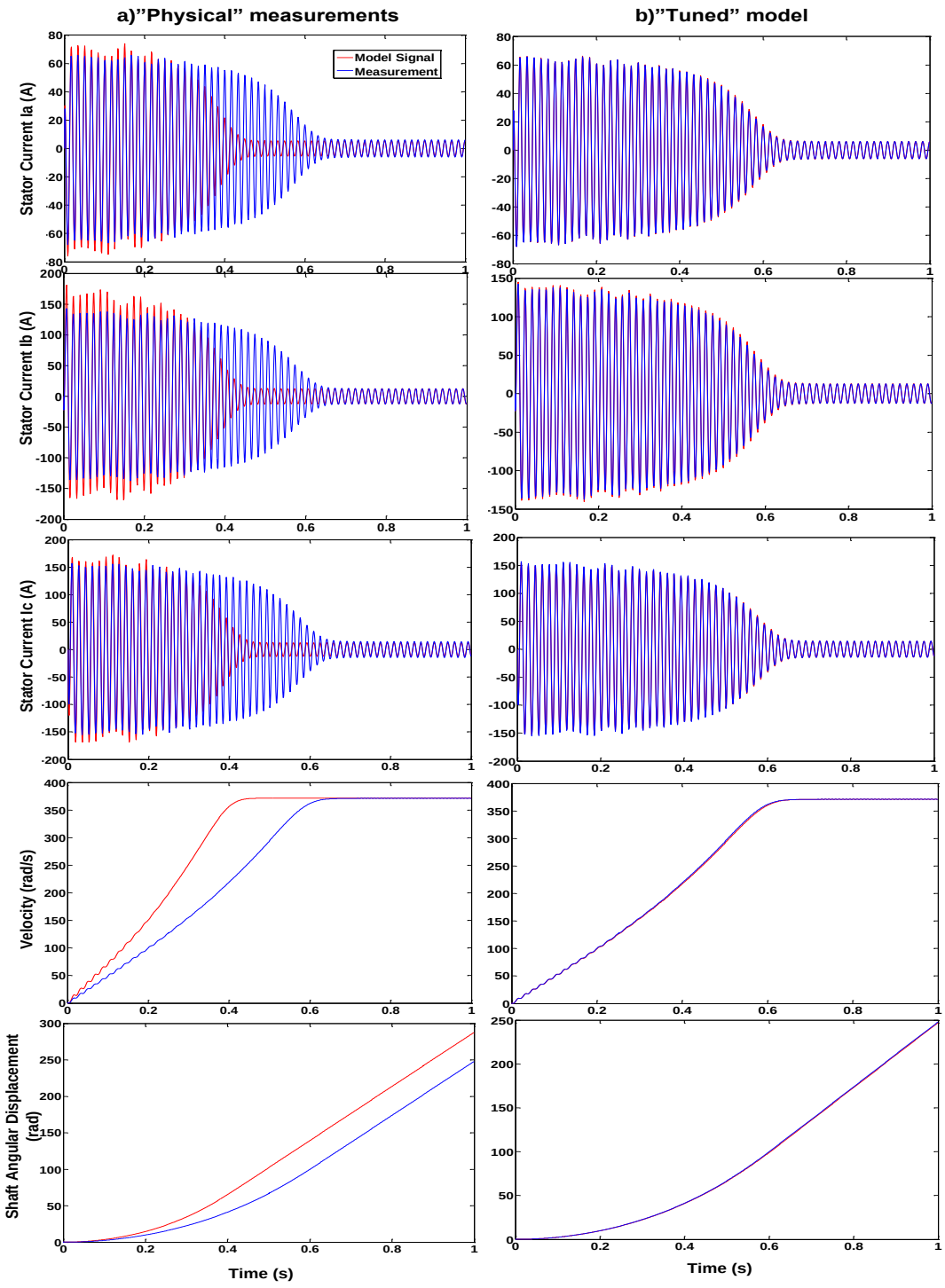

Figure 5.10: (a) Healthy & faulty system "measurements" (b) Comparison of model simulations with tuned parameters to "measurements" – Case 2

5.2.5 Case 3 – Resistance across all three phases doubled

The module was tested with all three phase resistances doubled. Results from the deterministic sampling algorithm are shown in Table 5.8.

Table 5.8: Top 5 possible points post Hammersley sampler – case 3

R_{sa} (Ω)	R_{sb} (Ω)	R_{sc} (Ω)	Cost function
3.6384	3.6022	5.5357	6.089006
4.3320	5.8000	2.7691	7.491397
4.3604	3.9113	4.4298	7.957445
3.2325	4.1645	5.2506	10.15588
4.3031	4.6297	3.6016	11.73476

Although the deterministic sampler results are inconclusive, the NSGA-II module receives well defined bounds for the global minimum search. Incorporating randomness in the search is effective only when the search region is compact. Otherwise, the algorithm will diverge from the global optimum. Results of the NSGA – II module shown in Table 5.9 reflect the true operating condition of the system, with resistances across all the three phases approximately doubled.

Table 5.9: Top 5 possible operating points after NSGA module – case 3

R_{sa} (Ω)	R_{sb} (Ω)	R_{sc} (Ω)	Cost function
4.27740	4.05089	4.29027	1.04682
4.27748	4.05060	4.29098	1.07832
4.27743	4.05089	4.29094	1.09164
4.27759	4.05087	4.29086	1.09470
4.27747	4.05086	4.29109	1.10052

Results for case 3 are shown in figure 5.11. Figure 5.11 is again similar to fig.5.9 where model simulations overlay measurements.

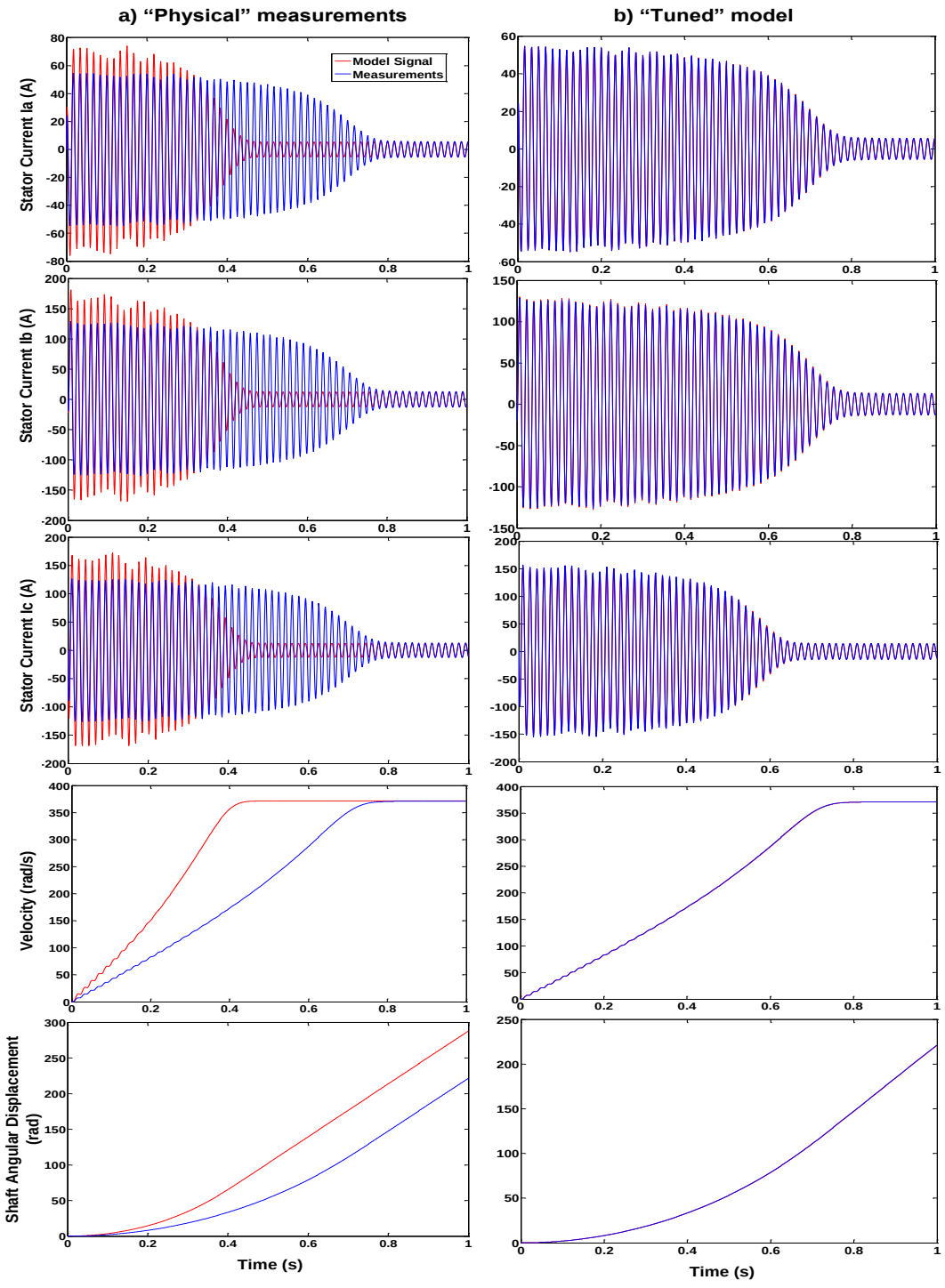

Figure 5.11: (a) Healthy & faulty system “measurements” (b) Comparison of model simulations with tuned parameters to “measurements” – Case 3

5.2.6 Case 4 – Resistance across two phases doubled, tripled across the third

The measurement model was adjusted to produce measurements corresponding to a fault with resistances across phases $'a'$ & $'c'$ doubled, and resistance of phase $'b'$ tripled. Results from the deterministic sampler module in table 5.10 are again inconclusive.

Table 5.10: Top 5 possible operating points post Hammersley sampler – case 4

R_{sa} (Ω)	R_{sb} (Ω)	R_{sc} (Ω)	Cost function
3.7424	5.2521	5.6727	4.836869
3.6956	7.0250	4.1863	8.208233
4.4695	5.7035	4.4777	9.074791
4.6589	4.8977	4.9564	9.822960
4.3401	6.2197	4.1849	11.50569

Results from the NSGA-II algorithm shown in table 5.11, have improved from the deterministic sampler output. The results in Table 5.11 initialized the parameter values for an extended Kalman filter (EKF) which can improve prediction accuracy by accumulating and using information on noise and system dynamics.

Table 5.11: Top 5 possible operating points after NSGA module – case 4

R_{sa} (Ω)	R_{sb} (Ω)	R_{sc} (Ω)	Cost function
4.31201	6.07756	4.27407	3.72625
4.39749	5.88988	4.34644	3.98757
4.38047	5.90344	4.34572	4.00057
4.38046	5.90386	4.34473	4.01827
4.38046	5.90307	4.34536	4.02041

The EKF is a sequential estimator unlike the deterministic sampler and NSGA – II module which operate on a given set of measurement data. Initial stages are important due to the high non-linearity of the model, coupled with divergence problems frequently encountered in EKF. Figure 5.12 shows results obtained after tuning the model parameters.

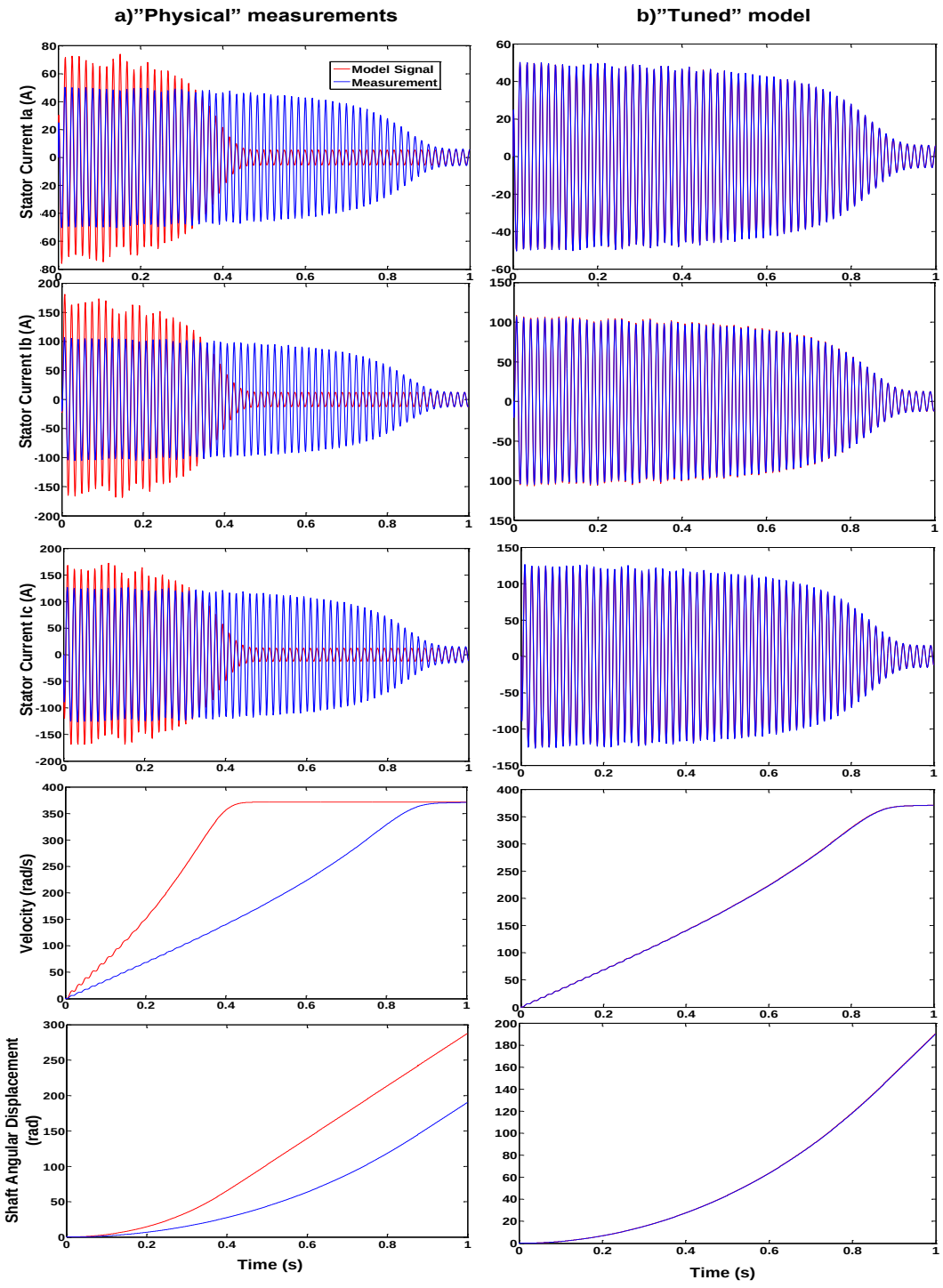

Figure 5.12: (a) Healthy & faulty system "measurements" (b) Comparison of model simulations with tuned parameters to "measurements" – Case 4

5.2.7 Algorithm and Computational specifications

Tuning module specifications and computational resources used are tabulated in Table 5.12. The module was run in MATLAB 2010a on a Linux platform.

Table 5.12: Algorithm & Computational Specifications for stage 1 & stage 2

Computational Specification	Value
Number of processor cores used for stage 1 (Hammersley sampler)	8
Number of processor cores used for stage 2 (NSGA – II)	5
Clock Speed of processor	3.324 GHz
Number of sample points for deterministic sampling	150
Number of generations for deterministic sampler	5
Population for genetic algorithm stage	50
Number of generations for genetic algorithm	3
Algorithm run time (deterministic sampler)	33 minutes
Algorithm run time (NSGA –II module)	180 minutes

5.2.8 Extended Kalman Filtering – Stage 3

The parameter values obtained from the Hammersley and NSGA-II stages initialized an extended Kalman filter (EKF). Figure 5.13 depicts the estimated states of the induction motor for case 1 – stator resistance of phase $'a'$ doubled.

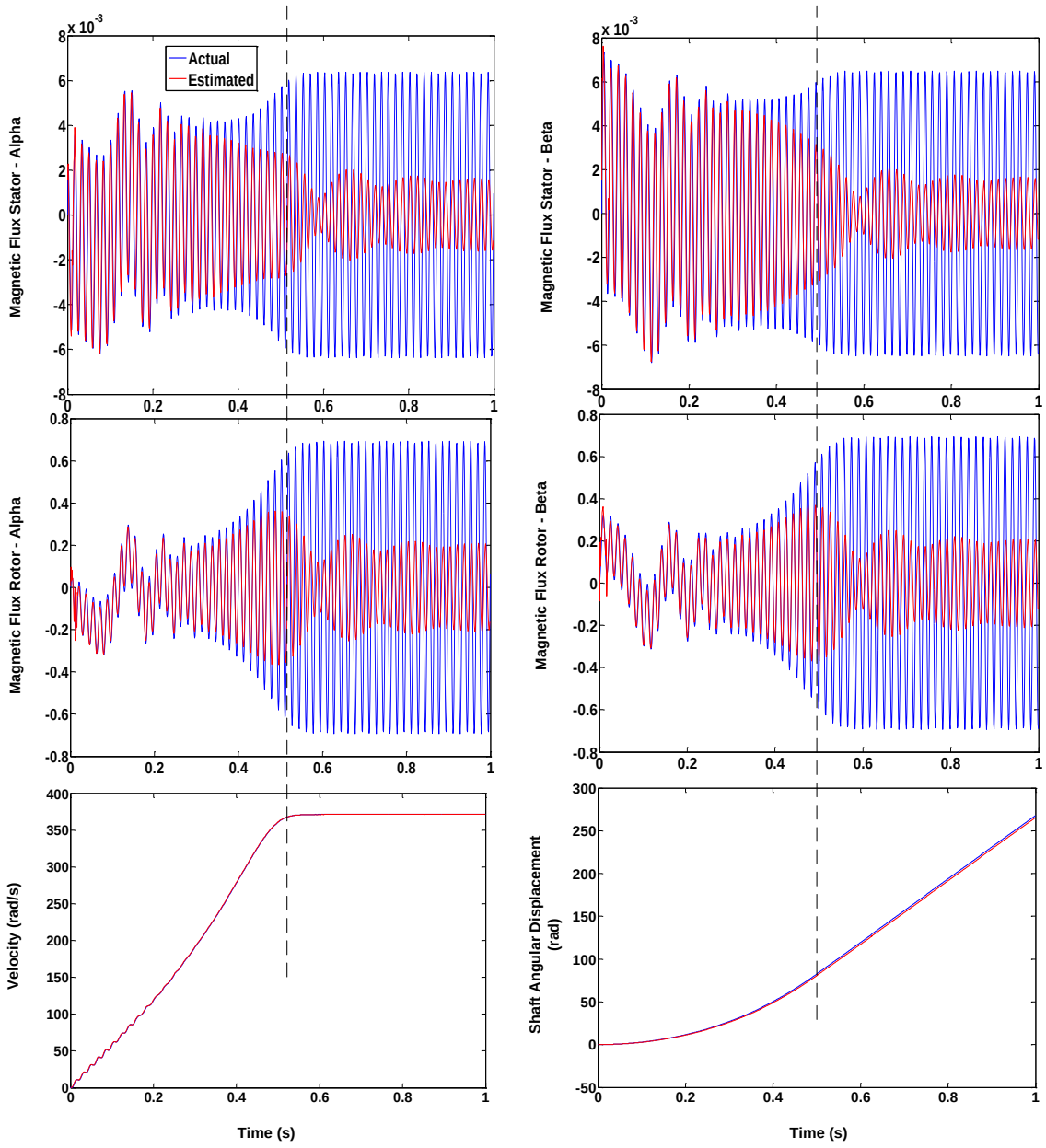

Figure 5.13: Estimation of states of induction motor using EKF – case 1

The vertical dashed line in figure 5.13 marks the transition from transient to steady state operating condition of the motor. The EKF filter performance beyond this transition is sub-optimal. An observability analysis on the system revealed that the system is not observable with the given set of measurements. The observability matrix was constructed using the MATLAB controls toolbox. Rank of the matrix was stored at every time step. Figure 5.14 plots the rank of the observability matrix versus time. The rank of the matrix at all time steps is less than 6 (total number of states), and fluctuates between 4 and 5.

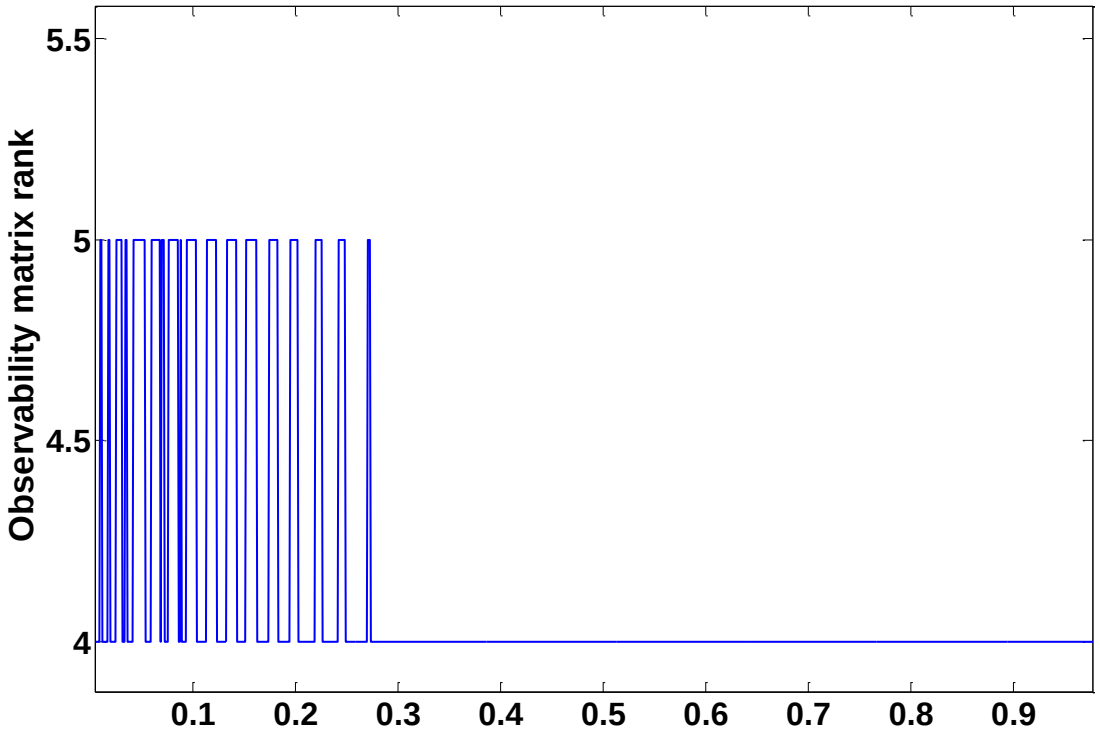

Figure 5.14: Rank of observability matrix at every time step

A “forgetting factor” of 0.9 was incorporated into the EKF and states were estimated. A forgetting factor forces the filter to give importance to incoming measurements. Figure 5.15 shows the estimated states using the modified EKF.

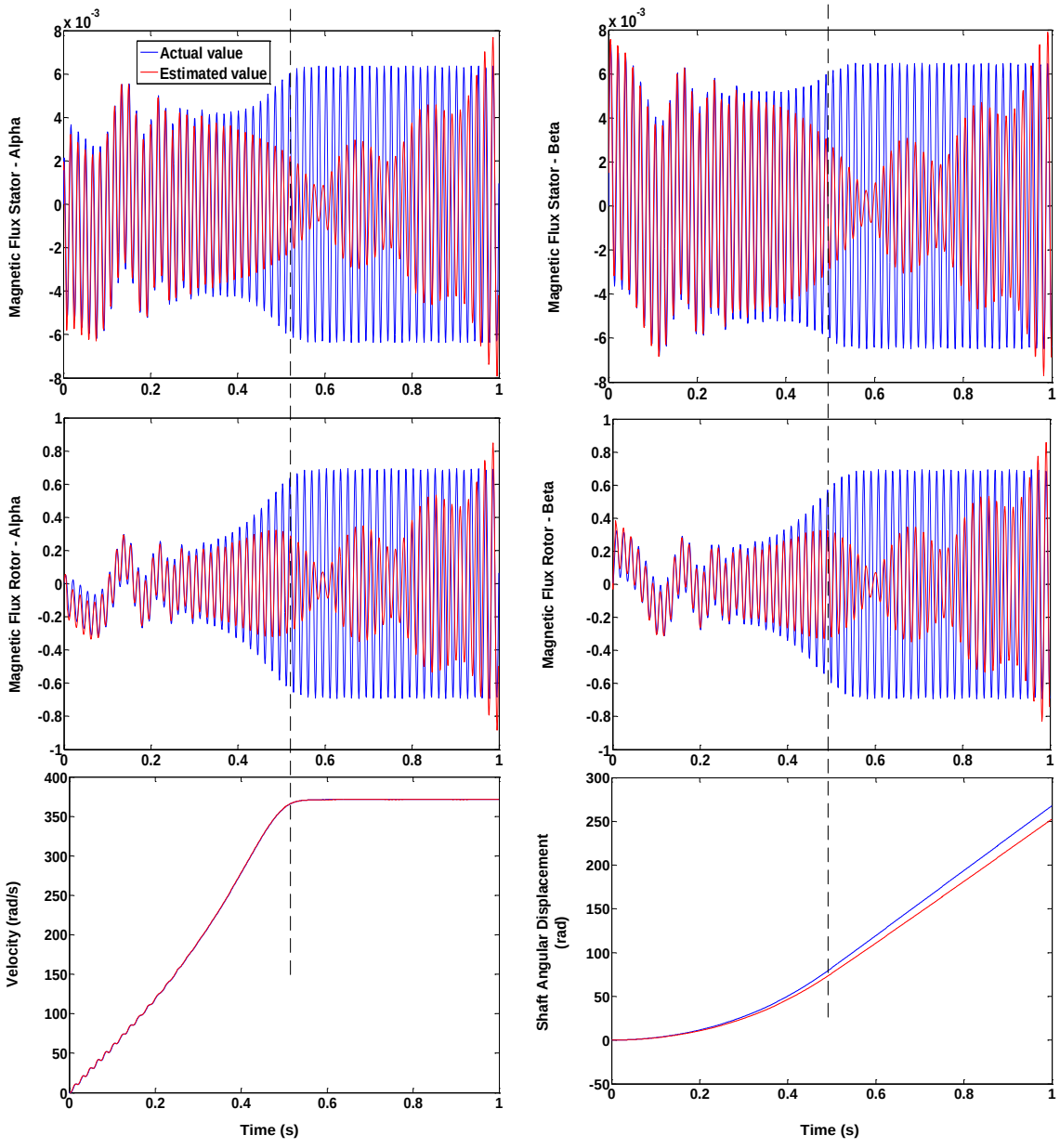

Figure 5.15: State estimation with ‘forgetting factor’ of 0.9

In figure 5.15, although the estimated states slowly catch up with the actual values in steady state condition, incorporating a 'forgetting factor' is not a reliable solution. In certain cases, it leads to filter divergence even in the transient operating condition. The stator resistance of phase $'a'$, estimated with and without forgetting factor is shown in figure 5.16.

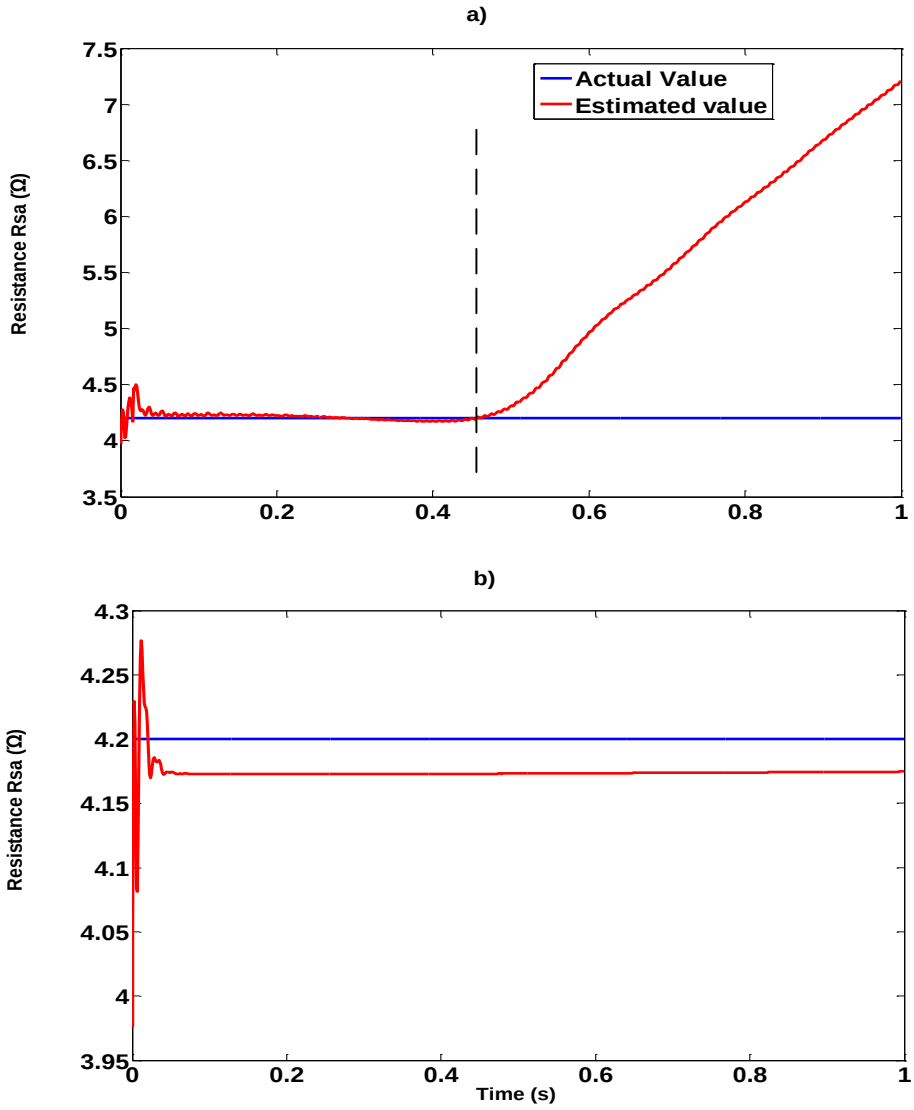

Figure 5.16: Parameter estimation (a) without forgetting factor (b) with forgetting factor

The vertical dashed line again marks the transition from transient to steady state. Filter divergence occurs once the system attains steady state. Surprisingly, the stator resistance estimation with a forgetting factor was more accurate. However a forgetting factor is not a generic solution, and will not work for many cases.

Analytical calculation of jacobians, for use in the EKF algorithm is not feasible for highly non-linear systems. Numerical differentiation was used to evaluate the process and measurement jacobians at every time step. Classical finite difference approximations for numerical differentiation are not very accurate [59]. The first derivative jacobians in this EKF implementation used the complex variable method given by the relation:

$$f'(x) = \frac{Im(F(x + ih))}{h} \tag{5.11}$$

where $'h'$ is the step size. The first derivative errors were in the order of 10^{-14}. Taking into account the reduction in computation time, the method was efficient and chosen for the jacobian evaluations in the EKF module.

To confirm the hypothesis of states and parameter estimation accuracy in the transient zone, the EKF was tested for case 4 – stator resistance of 2 phases doubled, and the third tripled. The filter was initialized using the parameter values obtained using Hammersley sampler and genetic algorithm stages. The magnitude of fault induced is higher and hence the time taken for the system to reach steady state is significantly higher compared to case 1. Figure 5.17 depicts the estimation of states for case 4. Forgetting factor was not used. It is again noted that filter performance deteriorates once the system reached steady state.

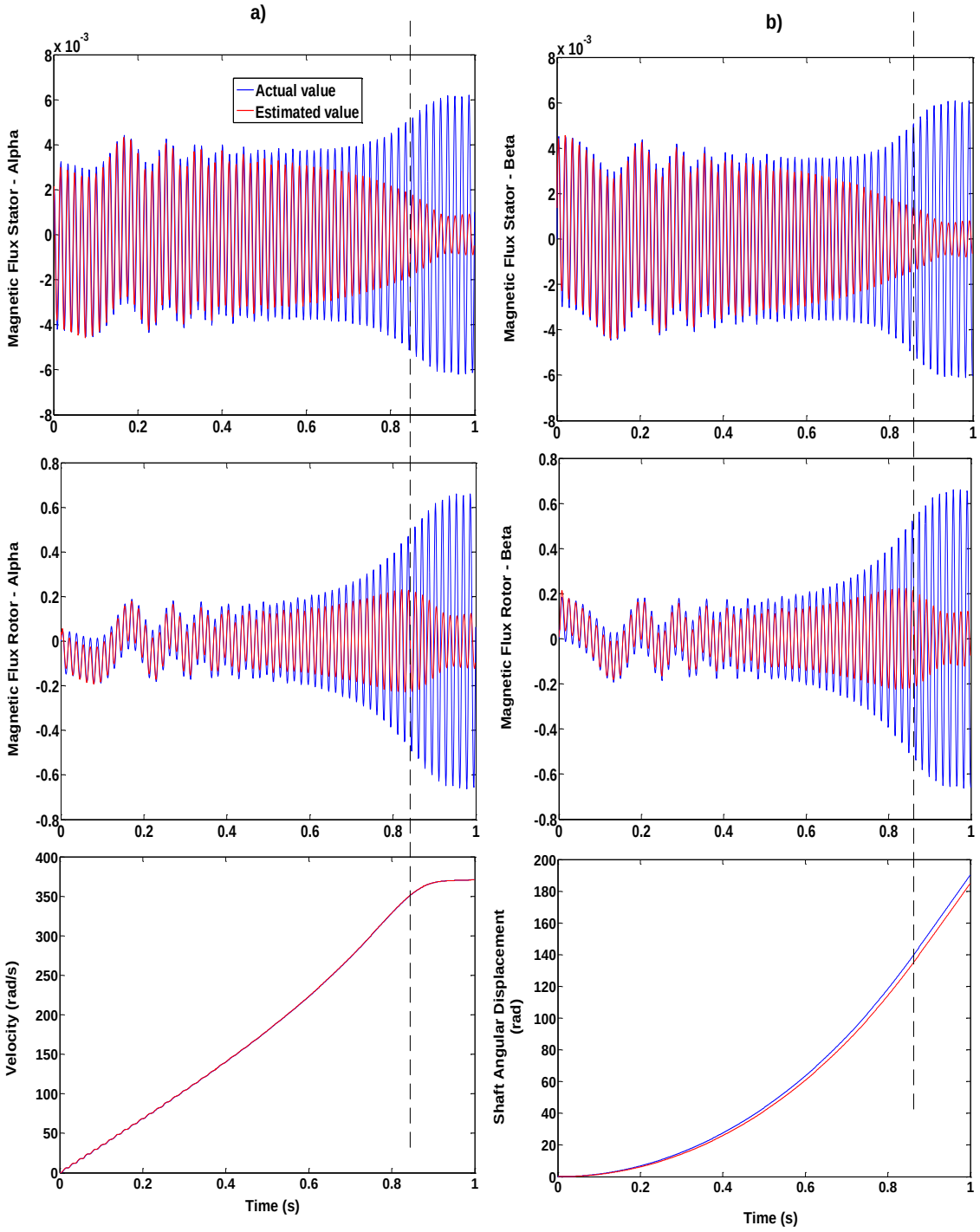

Figure 5.17: Estimation of states of induction motor using EKF – case 4

All three stator resistances were estimated and shown in figure 5.18. Filter divergence is evident at steady state.

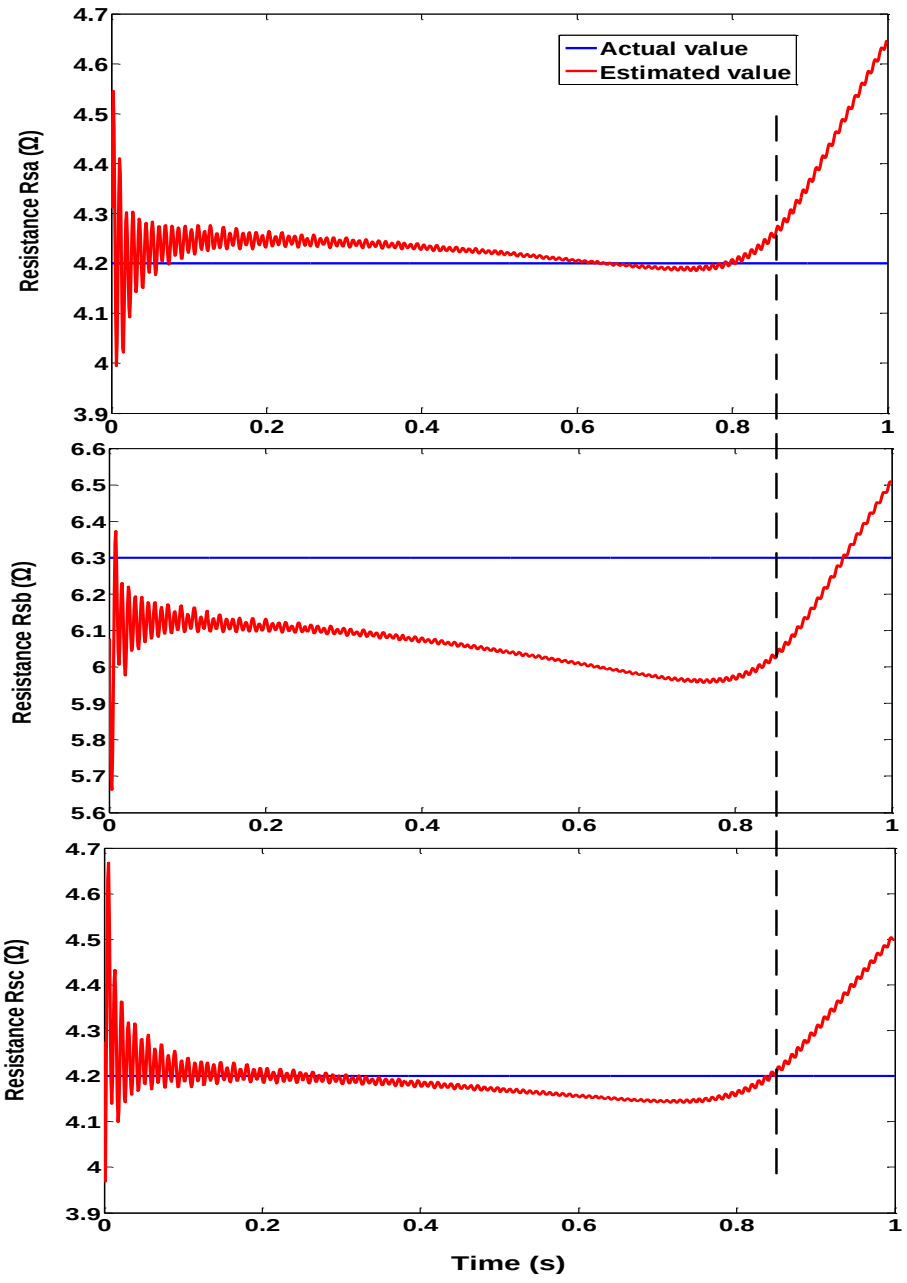

Figure 5.18: Estimation of three phase stator resistances of induction motor using EKF

5.2.9 Parameter Estimation at steady state using Hammersley and NSGA – Case 1

Measurements with the system operating at steady state were fed into the Hammersley and NSGA-II modules, to obtain parameter estimates of system at steady state. The outputs of the Hammersley and NSGA stages are tabulated in Table 5.13 and Table 5.14.

Table 5.13: Hammersley sampler results – case 1 (Steady State)

R_{sa} (Ω)	R_{sb} (Ω)	R_{sc} (Ω)	Cost function
3.7794	1.8078	2.8139	2.54178
4.1214	2.1205	1.8996	2.69270
4.0785	1.8679	2.9110	2.81606
3.9469	2.3586	1.9499	3.07824
2.9349	1.9852	4.0715	3.23118

Table 5.14: Top 5 operating points after NSGA module – case 1 (Steady State)

Case	R_{sa} (Ω)	R_{sb} (Ω)	R_{sc} (Ω)	Cost function
A	3.7794	1.8078	2.8139	2.54178
B	4.1214	2.1205	1.8996	2.69270
C	4.0785	1.8679	2.9110	2.81606
D	3.9469	2.3586	1.9499	3.07824
E	3.7329	1.8000	2.8856	3.12260

Table 5.14 shows the top 5 possible operating points of the system after results from the NSGA-II module was combined with the deterministic sampler results and sorted based on residual (cost function) values. Cases B and D firmly indicate doubling of phase $'a'$ resistance (4.2 ohm) and resistances across phases $'b'$and $'c'$around the nominal value (2.1 ohm). The deterministic sampler performed well for this case. Comparing the results in Table 5.13 and Table 5.14, it is noted that only the last row has changed which implies that the genetic algorithm stage could find only one solution with cost function lower than those of the deterministic sampler output.

5.2.9 Parameter Estimation at steady state using Hammersley and NSGA – Case 2

The measurement model was adjusted to simulate the induction motor output with resistance values of phases $'b'$and $'c'$ doubled. The output of the deterministic sampler module, when fed with data from the system operating at steady state, is shown in Table 5.15.

Table 5.15: Hammersley sampler results – case 2 (Steady State)

R_{sa} (Ω)	R_{sb} (Ω)	R_{sc} (Ω)	Cost function
2.5869	3.3300	4.7734	2.86163
2.2559	3.5632	4.7871	3.10530
2.9535	4.2230	3.2680	4.05263
2.8478	4.7836	2.7854	4.27125
2.1907	4.0709	4.2256	4.43678

Table 5.16: Top 5 operating points after NSGA module – case 2 (Steady State)

Case	R_{sa} (Ω)	R_{sb} (Ω)	R_{sc} (Ω)	Cost function
A	2.56816	5.07879	2.87877	2.60186
B	1.97716	3.89996	4.61514	2.70444
C	2.56797	5.07860	2.87987	2.73704
D	2.56797	5.07865	2.87746	2.82928
E	2.58690	3.33000	4.77340	2.86163

Cases B and E indicate doubling of resistances in phases $'b'$and $'c'$ and phase $'a'$ resistance around the nominal value. The other cases do not point to the exact operating condition. The other points are "local minima" discussed in Chapter 4 where completely different parameter combinations result in identical residual values. Still, two of the top 5 operating points obtained after the NSGA-II module reflect the true condition of the machine which is valuable information to the operator.

5.2.10 Parameter Estimation at steady state using Hammersley and NSGA – Case 3

The module was tested with all three phase resistances doubled. The output of the deterministic sampler module, when provided with data for the system operating at steady state, is shown in Table 5.16.

Table 5.16: Hammersley sampler results – case 3 (Steady State)

R_{sa} (Ω)	R_{sb} (Ω)	R_{sc} (Ω)	Cost function
4.4390	4.4312	3.6686	1.58202
4.2910	3.9119	4.0858	1.67222
4.7542	3.7931	4.2387	1.82793
3.5949	4.4524	4.5317	1.87469
4.4094	3.6349	4.6343	1.91185

Table 5.17: Top 5 operating points after NSGA module – case 3 (Steady State)

Case	R_{sa} (Ω)	R_{sb} (Ω)	R_{sc} (Ω)	Cost function
A	4.32690	4.20467	4.03964	0.506785
B	4.30258	4.16635	4.04572	0.551574
C	4.33596	4.18862	4.01482	0.597564
D	4.33543	4.18825	4.01457	0.598823
E	4.33448	4.18819	4.01448	0.599368

The output obtained after the NSGA-II module indicates that all three phase resistances have doubled in value (2.1 x 2=4.2 ohm). Looking at the cost function values in Table 5.17, the tuning module has performed exceedingly well for this case. This is a general trend observed in the parameter tuning module. When the phase resistances uniformly are increased or decreased across all the three phases, the tuning algorithm is very efficient in tracking those changes compared to the other cases that were tested.

5.2.11 Parameter Estimation at steady state using Hammersley and NSGA – Case 4

In the measurement model resistances of phases $'a'$and $'c'$ were doubled and phase $'b'$ resistance was tripled. The output of the deterministic sampler module, with the system operating at steady state is shown in Table 5.18.

Table 5.18: Hammersley sampler results – case 4 (Steady State)

R_{sa} (Ω)	R_{sb} (Ω)	R_{sc} (Ω)	Cost function
3.9375	5.7797	4.9378	1.94936
4.2142	3.8820	6.7753	2.20261
3.9333	3.9124	6.9658	2.55641
3.9895	4.8829	5.8742	2.56390
3.8576	4.7672	5.9188	2.57706

Table 5.19: Top 5 operating points after NSGA module – case 4 (Steady State)

Case	R_{sa} (Ω)	R_{sb} (Ω)	R_{sc} (Ω)	Cost function
A	3.9375	5.77970	4.93780	1.94937
B	4.2142	3.88200	6.77530	2.20262
C	4.1204	5.45007	5.09961	2.22260
D	4.7027	3.53933	6.65501	2.23948
E	4.7400	3.49609	6.66668	2.27514

The top 5 operating points of the system obtained after the NSGA-II module are shown in Table 5.19. The top point with the least cost function value (Case A) reflects the true operating condition of the system. Cases B, D and E support the proposition but with the phase resistances interchanged. The parameter combinations of cases B, D and E indicate that two of the phase resistances have doubled and one phase resistance has tripled. But contrary to the measurement model, these cases indicate phase resistances $'a'$ and $'b'$ have doubled and tripled across phase$'c'$. Though this is not the exact operating condition of the system, the values convey a similar interpretation which is valuable to the machine operator.

Chapter 6: Conclusion

A framework for non-linear parameter estimation in model based diagnostics has been presented. Though the module is yet to be tested on different types of machines and on real systems, initial results are encouraging.

6.1 Introducing the Graphical User Interface

A graphical user interface was built using MATLAB v 2010a and is shown in figure 6.1.

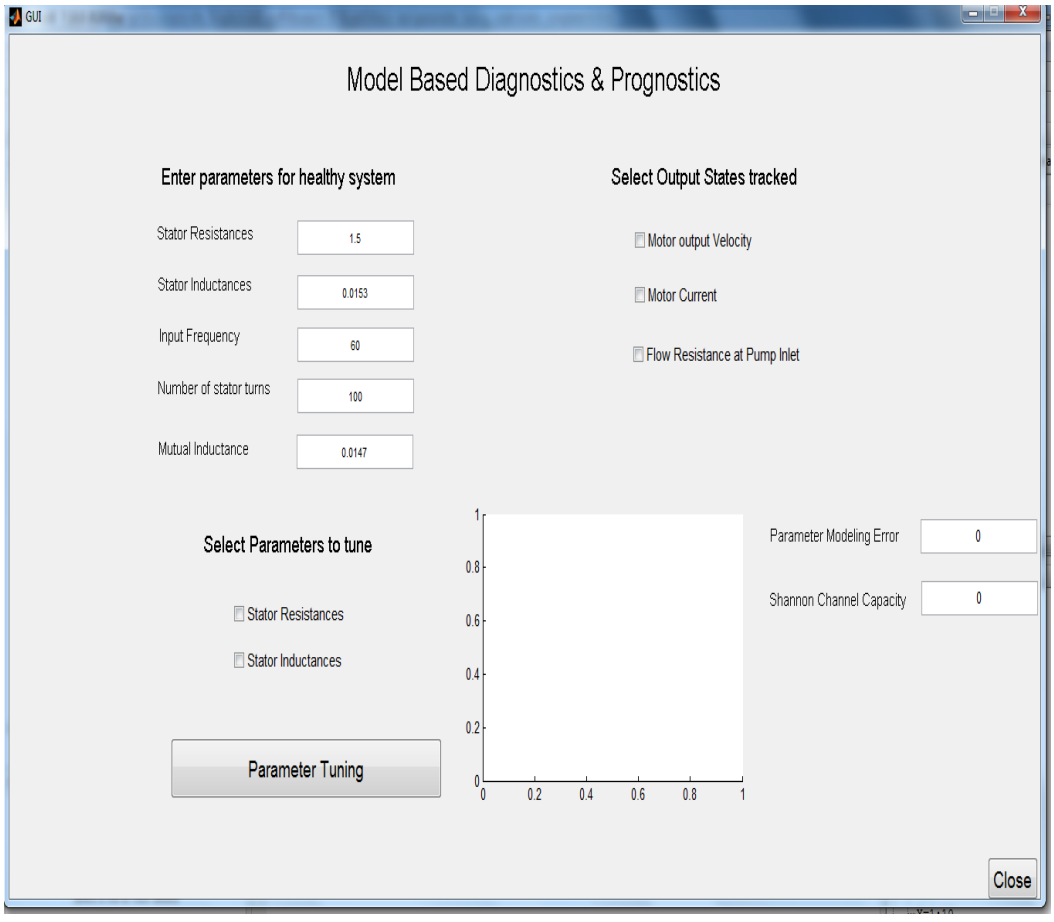

Figure 6.1: Graphical User Interface for induction motor diagnostics

The interface simplifies interaction between the diagnostic module and the operator. The operator can select sensor measurements. Based on the fault to be diagnosed, corresponding parameters can be chosen. Observability should be analyzed at this juncture to inform the operator whether the measurements contain enough information about states and parameters. Once the observability check is over, the tuning module is initiated. Bryant's [5] channel capacity theorem can be used as a health metric for the system and the value displayed in the 'Shannon Channel capacity' field. 'Modeling error' corresponds to the residual (or) cost function value after the parameters have been tuned. The plot in the interface tracks the channel capacity reduction over time and observes system performance. Once the channel capacity reduces below a threshold, maintenance is necessary. The interface is in a nascent stage. With more algorithms being developed, various functionalities can be added on to the module.

6.2 CONCLUSION

A framework for parameter tuning was established. Lessons learnt from this work include:

- The initial two stages – hammersley sampler and genetic algorithm (NSGA-II) are not as sensitive to observability issues as the Kalman filter. Even when fewer measurements were tracked and the system was not observable, the first two stages performed well for the induction motor case in transient as well as steady state operating conditions. The down-side is computational expense, and inability to estimate parameters online.

- Observability is essential for the extended Kalman filter stage, where the motive is to make the diagnostic module online. Update is based on incoming measurements.
- Building highly complicated models that try to capture every interaction inside the real system will help improve the performance of the initial two stages. But with the filter, simpler models tend to perform better. In the first two stages, the process flow is from state to measurement. Based on the states and parameter values, system response is simulated for each combination of parameters and compared with the actual measurements available. For filtering, the process is exactly reversed (Figure 4.7). Using system dynamics, the measurement is mapped back to individual states and parameters. If this mapping is highly non-linear, parameter estimation can become very challenging given the fact that parameter degradation dynamics is not available. Also, numerical errors might result due to jacobian calculations involved at every time step.
- Parameter estimation when the system is at steady state can be a challenge. In the induction motor case, the extended Kalman filter wouldn't work because the states were not observable through the measurements. Still during transient state, the filter performed fairly well because the measurement residuals (innovation) between a healthy and a faulty system were more pronounced and Kalman gain is large enough. An analogy can be constructed. When a car cruises on a highway and is subjected to a small excitation (say a bump), the faults become more evident – say a faulty bearing in one of the wheels or vibrations in the engine block. But when the car cruises at normal speed without any excitation, these faults are not as evident.

- With respect to selection of states, it was observed that stator current $'i'_a$ was extremely sensitive to the magnetic flux on stator alpha side $'\emptyset'_{\alpha s}$. A change as small as 10^{-10} in the state $'\emptyset'_{\alpha s}$ would affect the stator current $'i'_a$. This high sensitivity can make parameter estimation extremely difficult.
- By choosing system states that are directly measurable, the measurement jacobian matrices simplify to a one-one correspondence or a linear relationship. This helps in better parameter estimates.
- A more robust check for observability in nonlinear systems is by computing the observability gramians.

6.2 FUTURE WORK

There is no fixed procedure for parameter estimation, especially for non-linear systems. The rigorous framework presented maximizes accuracy of the estimation process. Model based diagnostics is promising because it is founded on sound engineering principles.

The module must be tested rigorously on real systems and on newly developed models. The parameter degradation process could be captured as differential equations with respect to the parameter and states using an entropy technique by Bryant [60] among others. The performance of the extended Kalman filter after incorporating these degradation dynamics should be studied and understood. Machine learning techniques can also be utilized to learn the relationship between parameters and system states. An advantage of a complex model is that simulation based testing, not possible through experimentation, can be done. Once the relationships are learnt, the performance of a filter can be tested based on these new dynamics.

Bibliography

[1] Isermann R., 2006, Fault-Diagnosis Systems -An Introduction from Fault Detection to Fault Tolerance.

[2] Committee in Analysis of Research Directions and Needs in U.S Manufacturing, 1991, The Competitive Edge: Research Priorities for U.S. Manufacturing.

[3] Isermann R., 1997, "Supervision, fault-detection and fault diagnosis methods - an introduction," Elsevier, **5**(5), pp. 639-652.

[4] Isermann R., 2005, "Model-based fault-detection and diagnosis – status and applications," Annual Reviews in Control, **29**(1), pp. 71-85.

[5] Bryant M. D., 2006, "Model- and Information Theory-Based Diagnostic Method," Journal of Dynamic Systems, Measurement, and Control, **128**(September), pp. 584-591.

[6] Kim J., and Bryant M., 2000, "Bond Graph Model of a Squirrel Cage Induction Motor With Direct Physical Correspondence," Journal of Dynamic Systems, Measurement, and Control, **308**(3), p. 205.

[7] Frankt P. M., 1990, "Fault Diagnosis in Dynamic Systems Using Analytical and Knowledge-based Redundancy A Survey and Some New Results," Control, **26**(3), pp. 459-474.

[8] Beard R., 1971, "Failure Accomodation in Linear Systems through Self-Reorganization."

[9] Jones H. L. E. E., 1973, "Failure Detection in Linear Systems."

[10] Clark R. N., 1975, "Detecting Instrument Malfunctions in Control Systems," IEEE Transactions on Aerospace and Electronic Systems, **AC-i 2**(4), pp. 412-473.

[11] Deckert J., 1977, "F-8 DFBW sensor failure identification using analytic redundancy," IEEE Transactions on Automatic Control, **22**(5), pp. 795-803.

[12] Isermann R., 1993, "Fault Diagnosis of machines via parameter estimation and knowledge processing," Automatica, **29**(4), pp. 815-835.

[13] Patton B. J., 1991, "Fault Detection and Diagnosis in aerospace systems using analytical redundancy," Computing & Control Engineering, **4138**(97).

[14] Willsky A. S., 1976, “A survey of design methods for failure detection in dynamic systems,” Automatica, **12**(6), pp. 601-611.

[15] Basseville M., 2000, “Subspace-based fault detection algorithms for vibration monitoring,” Automatica, **36**(1), pp. 101-109.

[16] Peuget R., 1998, “Fault detection and isolation on a PWM inverter by knowledge-based model,” IEEE Transactions on Industry Applications, **3**(6), pp. 281-1326.

[17] Filippetti F., 2000, “Recent developments of induction motor drives fault diagnosis using AI techniques,” IEEE Transactions on Industrial Electronics, **2**(6), pp. 1544-1004.

[18] Martins J. F., 2007, “Unsupervised Neural-Network-Based Algorithm for an On-Line Diagnosis of Three-Phase Induction Motor Stator Fault,” IEEE Transactions on Industrial Electronics, **2**(1), pp. 559-264.

[19] Isermann R., 1998, “On fuzzy logic applications for automatic control, supervision, and fault diagnosis,” IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans, **28**(2), pp. 221-235.

[20] Bishop R. R., and Richards G. G., 1990, “Identifying Induction Machine Parameters Using,” Generations Journal Of The American Society On Aging, pp. 476-479.

[21] Gertler J., 1997, “Fault detection and isolation using parity relations,” Control Engineering Practice, **5**(5), pp. 653-661.

[22] Mukund D., and Ray A., 1981, “A Fault Detection and Isolation Methodology,” IEEE, pp. 1363-1369.

[23] Chow E., 1984, “Analytical redundancy and the design of robust failure detection systems,” IEEE Transactions on Automatic Control, **29**(7), pp. 603-614.

[24] Lou X.-C., 1986, “Optimally robust redundancy relations for failure detection in uncertain systems,” Automatica, **22**(3), pp. 333-344.

[25] Mehra R., and Peschon J., 1971, “An Innovations approach to Fault Detection and Diagnosis in Dynamic Systems,” Automatica, **7**, pp. 637-640.

[26] Frank P. M., 1987, “Advanced Fault-detection and Isolation schemes using nonlinear and robust observers,” Proceedings of 10th IFAC Congress (Vol 4), pp. 63-68.

[27] Wilbers D. M., and Speyer J. L., 1989, “Detection filters for aircraft sensor and actuator faults,” ICCON ’89 International Conference on Control and Applications.

[28] Young P., 1981, “Parameter Estimation for continuous-time models - A survey,” Automatica, **17**(1), pp. 23-39.

[29] Cardozo E., and Taludkar S., 1988, “A Distributed Expert System for Fault Diagnosis,” IEEE Transactions on Power Systems, **3**(2), pp. 641-646.

[30] Comly J. B., and Dausch M. E., 1990, “Fuzzy Logic for Fault Diagnosis 1 INTRODUCTION,” Techniques, **1381**, pp. 390-400.

[31] Zamora J., and Cerrada A., 1998, “On-line estimation of the Stator Parameters in an Induction motor using Only Voltage and Current Measurements,” Control Engineering Practice, **6**(2), pp. 369-383.

[32] Said M. S. N., 2000, “Detection of broken bars in induction motors using an extended Kalman filter for rotor resistance sensorless estimation,” IEEE Transactions on Energy Conversion, **1**(5), pp. 251-70.

[33] Jack L. B., and Nandi A. K., 2002, “Fault Detection Using Support Vector Machines and Artificial Neural Networks, augmented by Genetic Algorithms,” Mechanical Systems and Signal Processing, **16**, pp. 373-390.

[34] Samanta B., 2003, “Artificial Neural Network based Fault Diagnostics of Rolling Element Bearings using Time Domain features,” Mechanical Systems and Signal Processing, **17**, pp. 317-328.

[35] Deb K., Agrawal S., Pratap A., and Meyarivan T., “A Fast Elitist Non-dominated Sorting Genetic Algorithm for Multi-objective Optimization : NSGA-II.”

[36] Unem R. K., Due P., and Vej J., 2003, “Parameter Identification of Induction Motors Using Differential Evolution,” Components, pp. 790-796.

[37] Bachir S., 2006, “Diagnosis by parameter estimation of stator and rotor faults occurring in induction machines,” IEEE Transactions on Industrial Electronics, **11**(2), pp. 62-973.

[38] Messaoudi M., Sbita L., and Abdelkrim M. N., 2007, “A robust nonlinear observer for states and parameters estimation and on-line adaptation of rotor time constant in sensorless induction motor drives,” International Journal, **2**(8), pp. 217-225.

[39] Giunta A., Wojtkiewicz S., and Eldred M., 2003, Overview of Modern Design of Experiments Methods for Computational Simulations.

[40] Woo T., 1995, "Efficient sampling for surface measurements," Journal of Manufacturing Systems, **14**(5), pp. 345-354.

[41] Kalagnanam J., and Diwekar U. M., 2010, "Efficient Sampling Technique for Quality Control," American Statistical Association and American Society for Quality, **39**(3), pp. 308-319.

[42] Park R., 1929, "Two-Reaction Theory of Synchronous Machines," Winter Convention of the A.I.E.E, New York.

[43] Ghosh B., and Bhadra S., 1993, "Bond Graph Simulation of a Current Source Inverter Driven Induction Motor (CSI-IM) System," Electrical Machines and Power Systems, **21**(1), p. 51.

[44] Singh G., 2002, "Multi-phase induction machine drive research—a survey," Electric Power Systems Research, **61**(2), pp. 139-147.

[45] Castanon L. E. G., and Oca S. M. D., Morales-Menendez, 2006, "Optimal Sampling for Feature Extraction in Iris Recognition Systems," MICAI 2006, LNAI 4293, pp. 810-819.

[46] Rafajlowicz E., and Schwabe R., 2006, "Halton and Hammersley sequences in multivariate nonparametric regression," Statistics & Probability Letters, **76**(8), pp. 803-812.

[47] Simpson T. W., Lin D. K. J., and Chen W., 2001, "Sampling Strategies for Computer Experiments: Design and Analysis," International Journal of Reliability and Applications, (814).

[48] Lee G., Mou J., and Shen Y., 1997, "Sampling Strategy Design for Dimensional Measurement of Geometric features using Coordinate Measuring Machine," Int. J. Mach. Tools Manufact., **37**(7), pp. 917-934.

[49] Srinivas N., and Deb K., 1994, "Muiltiobjective Optimization Using Nondominated Sorting in Genetic Algorithms," Evolutionary Computation, **2**(3), pp. 221-248.

[50] Deb K., Pratap a, Agarwal S., and Meyarivan T., 2002, "A fast and elitist multiobjective genetic algorithm: NSGA-II," IEEE Transactions on Evolutionary Computation, **6**(2), pp. 182-197.

[51] Chakraborty M., and Chakraborty U. K., 1997, "An analysis of linear ranking and binary tournament selection in genetic algorithms," Proceedings of ICICS, 1997 International Conference on Information, Communications and Signal Processing. Theme: Trends in Information Systems Engineering and Wireless Multimedia Communications (Cat. No.97TH8237), (September), pp. 407-411.

[52] Deb K., and Agrawal R. B., 1994, "Simulated Binary Crossover for Continuous Search Space," Mechanical Engineering.

[53] Maybeck P., 1979, "Stochastic models , Estimation and Control," New York, Academic Press.

[54] Gelb A., 1974, Applied Optimal Estimation, The Analytic Sciences Corporation.

[55] Kalman R. E., 1960, "A New Approach to Linear Filtering and Prediction Problems," ASME Journal of Basic Engineering, **82**(Series D), pp. 35-45.

[56] Welch G., and Bishop G., 2006, "An Introduction to the Kalman Filter," In Practice, pp. 1-16.

[57] Wan E. A., and Merwe R. V. D., 2000, "The Unscented Kalman Filter for Nonlinear Estimation," Adaptive Systems for Signal Processing, Communications and Control Symposium IEEE, pp. 153-158.

[58] Choi J., 2006, "Model based diagnostics of motors and pumps."

[59] Squire W., and Trapp G., 1998, "Using Complex Variables to Estimate Derivatives of Real Functions," Society for Industrial and Applied Mathematics, **40**(1), pp. 110-112.

[60] Bryant M., and Khonsari M., 2008, "Application of Degradation-Entropy Generation Theorem to Dry Sliding Friction and Wear," International Joint Tribology Conference, pp. 2008-2010.